TOWARDS 6G THROUGH SDN AND NFV-BASED SOLUTIONS FOR TERRESTRIAL AND NON-TERRESTRIAL NETWORKS

A Dissertation
Presented to
The Academic Faculty

By

Ahan Kak

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

May 2021

TOWARDS 6G THROUGH SDN AND NFV-BASED SOLUTIONS FOR TERRESTRIAL AND NON-TERRESTRIAL NETWORKS

Thesis committee:

Dr. Ian F. Akyildiz (Advisor)
School of Electrical and Computer Engineering (Formerly)
Georgia Institute of Technology

Dr. Chuanyi Ji
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Raghupathy Sivakumar (Chair)
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Henry L. Owen
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Mary Ann Weitnauer
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Andy Sun
School of Industrial & Systems Engineering
Georgia Institute of Technology

Date approved: April 6, 2021

To my family,

for their endless love, support, and encouragement.

ACKNOWLEDGMENTS

I would like to begin by expressing my deepest gratitude to my advisor, Dr. Ian F. Akyildiz. I am extremely grateful to him for giving me the life-changing opportunity of joining his lab. His unparalleled vision and boundless passion have been integral in setting me on the path to academic success. Like a guiding light that shines through stormy seas, Prof. Akyildiz's limitless wisdom has always led the way forward, even in the most difficult of times. His incredible work ethic has been a constant source of inspiration for me throughout this journey, one that has been greatly enriched by his immense knowledge and extensive experience. Being an international student is never easy, but Prof. Akyildiz's warmth, benevolence, and kindness are what made Atlanta home. I have been extremely fortunate to be advised in not just academics, but also life, by one of the greatest scientists of our generation, and for that I am forever indebted.

I would also like to express my profound appreciation to all the faculty members and administrative staffs of the School of Electrical and Computer Engineering at the Georgia Institute of Technology for their help throughout my Ph.D. A special thanks goes to Dr. Raghupathy Sivakumar and Dr. Mary Ann Weitnauer, who provided constructive and valuable suggestions on my research work and are a key part of my Ph.D. Defense Reading Committee. Moreover, I am thankful to Dr. Chuanyi Ji, Dr. Henry L. Owen, and Dr. Andy Sun, who are an integral part of my Ph.D. Defense Committee. Their highly insightful comments have helped me complete my Ph.D. degree and achieve a solid research path towards this thesis.

My sincere and warm thanks go also to the entire Broadband Wireless Networking Laboratory family including all current and former members, for their constant support and unwavering friendship throughout my years in the lab. A special mention goes to Bige Unluturk and Shuai Nie for the amazing times we shared during the course of our graduate years. A special thanks also goes to all the people I worked with as part of my research

projects. I would like to acknowledge the support of National Science Foundation through these projects, which made the completion of this thesis possible.

I am also grateful to Dr. Evgeny Khorov for hosting me in Moscow during the Summer of 2019 and Dr. Nakjung Choi for hosting me at Nokia Bell Labs during the Summer of 2020, and for their invaluable contributions to my growth as a researcher.

I would also like to thank all my friends and roommates through the years. They have been a integral part of this journey and I will forever cherish their support. And finally, last but by no means least, I can never find enough words to express my gratitude to my family, who have been my rock through this journey, especially my parents Vidyut and Kavita, my uncle Kowshik, and my aunt Hema. This achievement would have not been possible without their love, support, and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

As societal needs continue to evolve, there has been a marked rise in a wide variety of emerging use cases that cannot be served adequately by existing networks. For example, increasing industrial automation has not only resulted in a massive rise in the number of connected devices, but has also brought forth the need for remote monitoring and reconnaissance at scale, often in remote locations characterized by a lack of connectivity options. Going beyond 5G, which has largely focused on enhancing the quality-of-experience for end devices, the next generation of wireless communications is expected to be centered around the idea of "wireless ubiquity". The concept of wireless ubiquity mandates that the quality of connectivity is not only determined by classical metrics such as throughput, reliability, and latency, but also by the level of coverage offered by the network. In other words, the upcoming sixth generation of wireless communications should be characterized by networks that exhibit high throughput and reliability with low latency, while also providing robust connectivity to a multitude of devices spread across the surface of the Earth, without any geographical constraints.

The objective of this PhD thesis is to design novel architectural solutions for the upcoming sixth generation of cellular and space communications systems with a view to enabling wireless ubiquity with software-defined networking and network function virtualization at its core. Towards this goal, this thesis introduces a novel end-to-end system architecture for cellular communications characterized by innovations such as the AirHYPE wireless hypervisor. Furthermore, within the cellular systems domain, solutions for radio access network design with software-defined mobility management, and containerized core network design optimization have also been presented. On the other hand, within the space systems domain, this thesis introduces the concept of the Internet of Space Things (IoST). IoST is a novel cyber-physical system centered on nanosatellites and is capable of delivering ubiquitous connectivity for a wide variety of use cases, ranging from monitoring and reconnaissance

to in-space backhauling. In this direction, contributions relating to constellation design, routing, and automatic network slicing form a key aspect of this thesis.

CHAPTER 1
INTRODUCTION

Wireless communication systems have experienced substantial revolutionary progress over the past couple of years, driven in large part by advancements within the domains of cellular systems and the Internet of Things (IoT). Numerous burgeoning applications and verticals, including virtual and augmented reality, e-commerce, contact-less payment, machine-to-machine communications, and enhanced mobile broadband, among others, have demonstrated the vast potential of 5G, which continues to evolve and adapt to a wide variety of emerging use cases.

However, as societal needs continue to evolve, there has been a marked rise in a wide variety of emerging use cases that cannot be served adequately by existing networks. For example, increasing industrial automation and the move from Industry 4.0 to the upcoming Industry $\mathrm{X}.0$ paradigm has not only resulted in a massive rise in the number of connected devices, but has also brought forth the need for remote monitoring and reconnaissance at scale. Furthermore, as shown in Figure 1.1, the proliferation of smart infrastructure and connected supply chains has led to the need for reliable remote connectivity. A key component of reliable connectivity is network resilience, i.e., the ability to maintain connectivity even when the ground infrastructure is subject to damage.

Consequently, "wireless ubiquity" has emerged as a central theme of the upcoming sixth generation of wireless networks, i.e., the quality of connectivity is intrinsically tied to the level of coverage provided by the network. Simultaneously, from a network architect's perspective, the network must exhibit a high level of scalability and flexibility while requiring low CAPEX and OPEX. To this end, software-defined networking (SDN) and network function virtualization (NFV) have been widely recognized as key enabling technologies for 5G and 6G communication systems [1]. While SDN seeks to separate the control plane

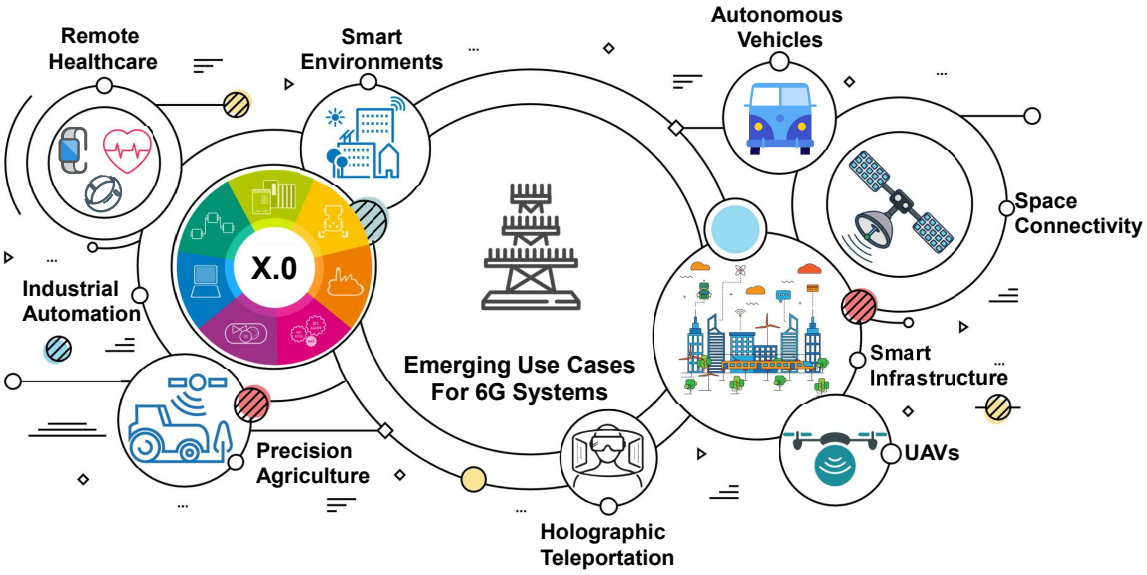


Figure 1.1: Emerging use cases for 6G systems.

from the data plane and introduce novel network control functionalities based on an abstract representation of the network, NFV decouples specific network functionality from specialized hardware, and implements the same via software on whitebox hardware. A combination of SDN and NFV vastly simplifies network deployment and operation, and enables independent evolution of both hardware and software, allowing for flexible adoption of emerging technologies in both domains. In order to achieve this vision of wireless ubiquity, in addition to cellular networks which are vital for robust terrestrial connectivity, nanosatellite systems are critical for enhancing coverage at low costs. Furthermore, SDN and NFV are also equally necessary for establishing networks that are easy to deploy, manage, and operate. In this thesis, we establish the architectural foundations for SDN and NFV-based cellular and space communications systems, which, as shown in Figure 1.2, are expected to be among the key enabling technologies for 6G [1].

System architecture design presents unique sets of challenges within two contexts– cellular networks and satellite systems. While the cellular systems landscape has evolved over multiple generations with a plethora of architectures and protocols, nanosatellite systems are still in their infancy. Consequently, the challenges associated with the former relate to the

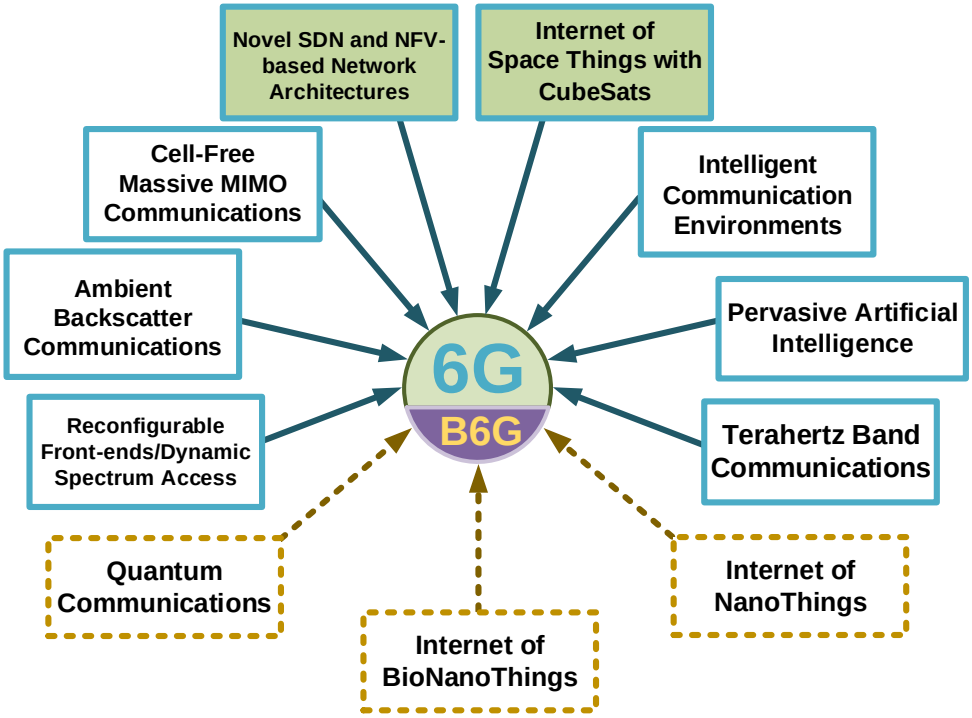


Figure 1.2: Key enabling technologies for 6G and beyond wireless communications systems.

enhancement of established paradigms, while the latter requires the design of new solutions from the ground up.

In particular, as part of this thesis, we introduce a novel network architecture for end-to-end cellular communications, along with solutions for radio access (RAN) and core network design. Furthermore, within the satellite systems domain, we propose and develop a new cyber-physical system centered on nanosatellites (CubeSats), known as the Internet of Space Things (IoST), along with key contributions to system constellation design, routing, and network slicing within IoST.

1.1 Key Themes and Research Objectives

In line with the notion of wireless ubiquity, this thesis spans two major directions: (i) flexible network architectures for 5G and beyond cellular systems; and (ii) the Internet of Space Things with CubeSats, each with a set of research objectives catering to the realization of novel networking paradigms.

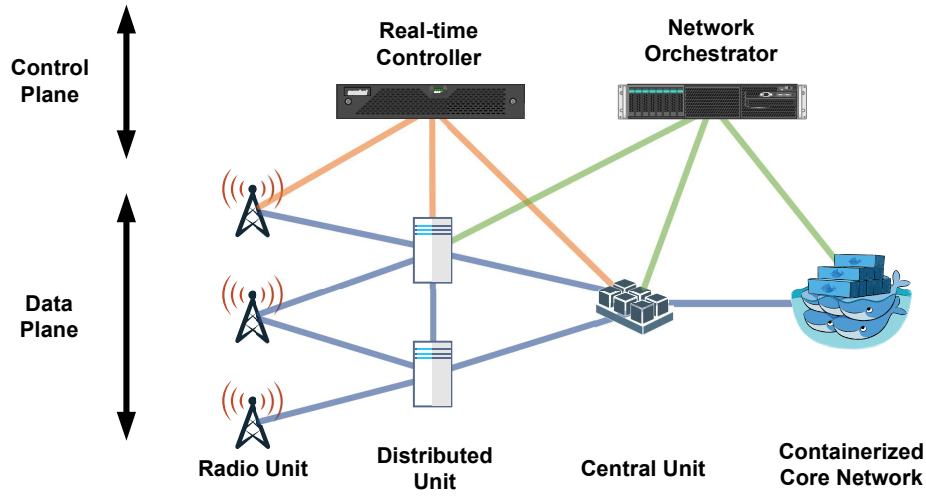


Figure 1.3: General overview of an SDN and NFV-based cellular network.

1.1.1 Flexible Network Architectures for 5G and Beyond Cellular Systems

System Architecture Design [2] The expansion of the service scope of cellular networks to include a wide variety of services such as mobile broadband and mission-critical machine-type communications has significantly shaped the evolution towards 5G and beyond systems. All these services impose divergent and often mutually exclusive requirements in terms of data rate, latency, and energy efficiency. To satisfy heterogeneous requirements, 5G and beyond systems should imbibe properties such as quality-of-experience awareness, adaptability and flexibility, scalability and reliability, support for multiple radio access technologies, and backward compatibility, all at a low CAPEX and OPEX. In recent years, a plethora of SDN-based cellular network architectures have been introduced, each with their own unique features and drawbacks. In general, an SDN and NFV-based architecture emphasizes the use of commodity computing hardware and espouses a separation between the data and control planes, as shown in Figure 1.3. Within this context, this research objective introduces a new cellular system architecture called ARBAT. ARBAT is characterized by many innovative features such as the Universal Network Device and Unified Cellular Network concepts, multi-slice modular resource management with the AirHYPE wireless hypervisor, network-user application interaction through the xStream platform, and simplified

multi-tenant orchestration through ServiceBRIDGE. A qualitative evaluation and feature comparison of ARBAT with other state-of-the-art architectures has also been conducted to demonstrate that ARBAT satisfies the aforementioned objectives of 5G and beyond systems.

Radio Access Network Design with Software-Defined Mobility Management [3] The softwarization of wireless networks has necessitated an overhaul of existing mobility management strategies. Specifically, mobility management is no longer constrained to function within the boundaries of a statically defined radio access network (RAN). Softwarization of the network infrastructure allows resource configurations and associations to be changed on demand, in a manner so as to support a least cost mobility management framework. To this end, this research objective presents an optimal RAN design framework augmented with user-specific clusters from the perspective of mobility management. The proposed framework is supported by a detailed mathematical model that characterizes user mobility, system traffic, and signaling costs. Performance evaluation is based on a cost comparison with conventional LTE/NR networks, and reinforces the fact that the framework proposed herein results in a significant reduction in signaling costs, even in the face of changing network scenarios.

Adaptive Containerization for Microservices-Based Core Networks [4] The core networks for 5G and beyond systems will be based on 3GPP's Service-Based Architecture (SBA). SBA mandates that the cellular core must consist of a standardized set of interconnected network functions that can exchange information over well-defined APIs. At the same time, in the interest of increased flexibility and improved scalability, each core network function can be further decomposed into a set of microservices amenable to containerization. However, while container-based virtualization is the preferred method for microservice deployment, communications service providers have not had much opportunity for cost and resource optimization. To address this issue, this research objective introduces a robust resource allocation framework for the containerized deployment of microservices across

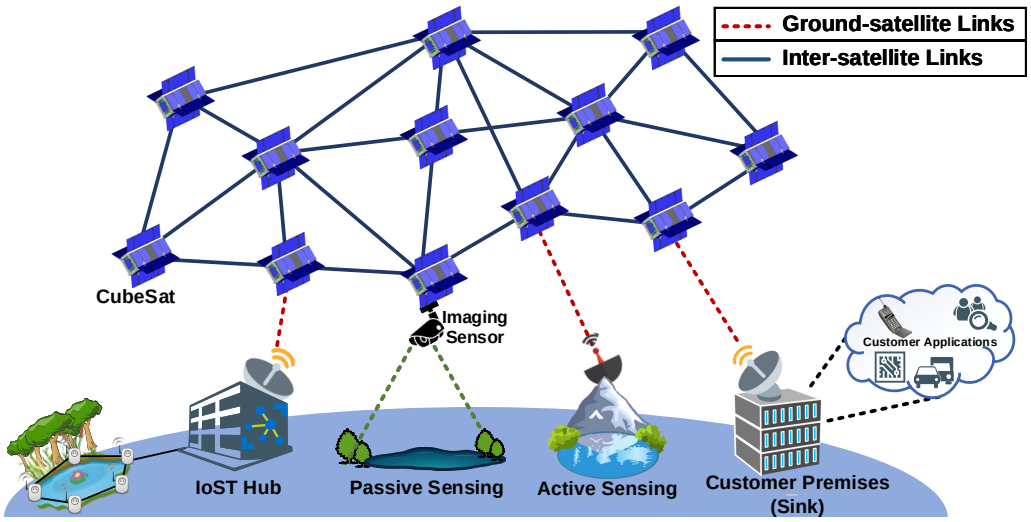


Figure 1.4: System overview of the Internet of Space Things with CubeSats.

distributed cloud systems. The proposed framework has been designed to reduce operating costs while guaranteeing service-level agreement (SLA) enforcement. The resulting performance evaluation demonstrates that the developed framework is able to achieve a significant reduction in system resource utilization when compared to state-of-the-art deployment strategies, while allowing for increased reliability.

1.1.2 The Internet of Space Things with CubeSats

End-to-End System Design [5, 6, 7] While the IoT market continues to grow at an exponential pace, the long term success of IoT is tied to its pervasiveness, an area where the heterogeneous connectivity solutions of today fall short by a large margin. The true potential of IoT can only be realized when it is augmented with a ubiquitous connectivity platform capable of functioning even in the most remote of locations. In this research objective, a novel cyber-physical system spanning ground, air, and space, called the Internet of Space Things (IoST) is introduced, as shown in Figure 1.4. Centered around nanosatellites known as CubeSats, IoST is envisioned as a means to achieving global connectivity at low costs, which is further bolstered by the use of SDN and NFV which provide fine-grained control over the system hardware, improve network resource utilization, and simplify network

management. In addition to a detailed system description, novel solutions for tackling peculiarities of the space environment are also provided. Furthermore, the system's potential is showcased through a set of preliminary performance evaluation metrics.

Large-Scale Constellation Design for Ultra-Dense CubeSat Networks [8, 9] IoST is expected to serve a wide variety of applications ranging from monitoring and reconnaissance to in-space backhauling. The pervasiveness of cyber-physical systems of this kind necessitates a robust constellation design, characterized by optimized coverage and consistent connectivity. Thus, optimal constellation design is of great importance to system architects. However, the constellation design frameworks prevalent today do not scale well beyond a few dozen satellites, adversely impacting the system's operational abilities. To this end, the objective of this research is two-fold. The primary objective is the development of a modular and highly customizable large-scale constellation design framework, with the secondary objective involving the use of the proposed framework for designing robust constellations for IoST, serving a wide variety of use cases ranging from global coverage to region-specific coverage scenarios. More specifically, the framework presented herein has been developed to optimize constellation design based on CubeSat density, as well as a rigorous mathematical characterization of coverage and connectivity parameters. Furthermore, an extensive set of performance comparisons with existing state-of-the-art constellations has been presented to validate the efficacy of the IoST constellations designed using the proposed framework. Finally, the impact of design parameter variations on the developed constellations has also been examined in great detail.

Online Segment Routing for Software-Defined CubeSat Networks [10] The increasing popularity of CubeSats has given rise to the possibility of ubiquitous cyber-physical systems serving a wide variety of applications. Expectedly, such systems are characterized by the need for a robust data routing framework that is purpose-built for resource-constrained CubeSats. To this end, an SDN-based segment routing (SR) framework for CubeSats has

been introduced as part of this research objective. More specifically, a robust analytical characterization of the SR problem has been presented, along with an online algorithm for near-optimal route computation characterized by a provable performance bound. Furthermore, a comprehensive performance evaluation has also been provided, with the results obtained being benchmarked against classical SDN systems. The results demonstrate that the proposed framework not only ensures a higher level of demand satisfaction but also results in a significant reduction in control traffic, along with load balancing.

Automatic Network Slicing for Space-Ground Integrated Networks [11] Taking into consideration the use cases associated with IoST, the network architecture must serve a plethora of application scenarios with differing SLA requirements over the same physical infrastructure in an end-to-end manner. At the same time, since the different use cases might belong to a variety of different stakeholders, IoST must support multi-tenancy and functional isolation of services. Consequently, a network slicing framework is vital to the success of IoST in particular, and space-ground integrated networks (SGINs) in general. To this end, an automatic network slicing framework for SGINs is presented as part of this research objective. The proposed framework has been designed to address the dual objectives of route computation and resource allocation with minimal SLA violations. Different from the existing state-of-the-art, the framework presented herein is purpose-built for ultra-dense SGINs, and is fully automated. In other words, the framework is purely SLA-based, and does not require prior information concerning the resource requirements associated with a slice. Other key innovations introduced through this framework include a robust SLA model for slice customization, a novel topology construction mechanism, and a unique segment routing-based online admission control solution. Furthermore, the flexibility and efficacy of the proposed framework has been evaluated through a comprehensive use case driven evaluation scenario.

1.2 Organization of the Thesis

The rest of this thesis is organized as follows. An overview of the existing body of work on SDN and NFV solutions for cellular systems and satellite networks is given in Chapter 2. Then, the proposed end-to-end novel network architecture for cellular systems is presented in Chapter 3, followed by the RAN design framework augmented with software-defined mobility management in Chapter 4, accompanied by the core network containerization solution in Chapter 5. Within the satellite systems domain, the IoST system architecture is presented in Chapter 6. Then, in Chapter 7, the large-scale constellation design problem is presented, highlighting the need for a robust constellation design framework to support the optimal operation of IoST. In Chapter 8, software-defined segment routing is proposed as a solution to combat the proliferation of control traffic in long fat networks such as IoST, thereby greatly improving network performance. Furthermore, optimal space-ground integration is achieved through the automatic network slicing solution presented in Chapter 9. In Chapter 10, the research contributions are summarized and future research avenues are discussed.

CHAPTER 2
PREVIOUS WORK

This chapter of the thesis contains a review of the literature for the related research on SDN and NFV-based solutions for cellular and space communications systems. This review is organized as follows. In Section 2.1, current state-of-the-art SDN-based cellular network architectures are discussed along with the standardization efforts in this domain, while Section 2.2 contains an overview of the research efforts concerning existing and upcoming CubeSat-based IoT and broadband solutions within both the industry as well as academia.

2.1 Software-Defined Mobile Network Architectures

In recent years, the domain of Software-Defined Mobile Networks (SDMNs) has witnessed much traction from both academia and industry. Early efforts in the SDMN domain were based on the softwarization of the core network (CN) [12], however the focus has shifted to the RAN [13, 14] in the past years. Today, there exists an exhaustive body of work on 5G and beyond wireless network architectures. In this section, we discuss the key works highlighting their unique features and primary drawbacks.

While standards pertaining to SDN-based cellular networks are still in their infancy, the 3GPP Control and User Plane Separation (CUPS) [15] paradigm introduced in 3GPP Release 14, is partially based on SDN principles. At the same time it is important to note that CUPS is patently different from SDN by design, with the 3GPP control (3GPP-C) and user (3GPP-U) planes being different from the SDN control (SDN-C) and data (SDN-D) planes. First, CUPS splits the data path into two paths, namely, the control traffic data path and user traffic data path. The 3GPP-C includes all functions that deal with the control traffic data path, while the 3GPP-D consists of the functions that process user traffic towards the data network. In other words, both the control and data plane functions take part in

packet forwarding. This approach is in contrast to SDN which mandates that all network traffic must pass through the SDN-D, with the SDN-C having no role in the physical packet forwarding action. Second, as of the Release 14 CUPS is limited to the CN only, with a partial separation of the RAN under study [16], i.e., a large portion of the RAN still follows the classical coupled control and user plane paradigm. However, a proper realization of the SDN concept requires complete separation of the entities that take the network control decisions from those that apply such decisions.

A monolithic 5G architecture based on SDN and NFV concepts is introduced in [17]. The principal advantage of this architecture is the cluster-based RAN concept with local controllers responsible for each cluster, allowing for efficient content caching, and inter-cluster scalability. However, we note that in the presented architecture, each local controller itself is a physical network function (PNF) with fixed functionality. Furthermore, while the overall network is scalable at the cluster level because of cluster independence, scalability within a cluster is a challenge, due to the potential bottleneck arising at a given local controller as the cluster size grows.

In a similar manner, SoftNet [18] considers the base station as a PNF, and proposes a unified RAN and an SDN-based CN. Additional features include a unified Radio Resource Management (RRM) framework and QoS mapping. While the control plane is distributed, it is primarily deployed within the CN, and the CN edge with a static distribution, i.e., functions cannot be moved across network elements, and therefore the flexibility suffers. Moreover, the unified resource scheduler cannot satisfy delay requirements for low-latency traffic. FlexRAN [14] is one of the few RAN solutions to present a functional proof of concept that includes the RAN controller, and Northbound (NB) and Southbound (SB) APIs. However, FlexRAN also considers a monolithic base station which cannot adapt to different use-cases. Further, the implementation is limited to LTE-based access and includes only one use-case based evaluation.

A classical SDN architecture using the OpenDaylight (ODL) platform is introduced

in [19]. The accompanying prototype is based on the ODL controller and emulated base stations. The developed framework is validated with enhanced Inter-Cell Interference Coordination (eICIC) deployed as a control function. The drawback of this architecture is the lack of flexibility. The RAN has only one functional split, and the network functions cannot be moved from one network node to another.

The vRAN Fronthaul group of the Telecom Infra Project (TIP) provides a virtualized base station solution that includes a virtualized Baseband Unit (vBBU), and a Remote Radio Unit (RRU) [20] . The highlight of the vRAN solution is its ability to function across a variety of fronthaul options using multiple physical layer functional splits, and fronthaul bandwidth compression. However, since vRAN caters to physical layer functional splits only, many network functions are deployed in a centralized manner at the vBBU. While this approach is useful for features such as centralized scheduling, and coordinated multipoint (CoMP), it also renders the architecture inflexible in the face of low latency use-cases that mandate a distributed function deployment close to the end-users. Furthermore, vRAN provides an NFV solution only and does not incorporate SDN, i.e., control and data plane separation is absent.

Similar to TIP, NEC also provides a virtualized C-RAN Distributed Unit (DU) solution [21], that supports two distinct functional splits between the DU and Radio Unit (RU) – the L2 and L3 splits. In the L2 split, physical layer functions are deployed at the RU, whereas in the L3 split, the RU additionally performs the MAC and RLC functions. On the one hand, the L2 split offers greater centralization, while on the other hand, the L3 split is better suited for low latency use-cases. In that sense, NEC's solution is more versatile but less granular than TIP's. However, despite the perceived versatility, NEC's architecture does not make use of SDN, and is an NFV-only solution at the moment.

Very recently, the xRAN forum [22] merged with the C-RAN alliance to form the ORAN alliance. More specifically, the alliance provides a RAN architecture [23] and a fronthaul specification [24]. Supporting both SDN and NFV, ORAN introduces a standalone ORAN

controller that interacts with the Central Unit (CU) and Distributed Unit (DU) to optimize overall network performance. ORAN also includes a relational database that reflects the network's state and is used by the network controller to carry out its operations. However, the current specification provides support for a single functional split only that concentrates much of the RAN functions at the CU. This can potentially decrease the performance in scenarios with several types of traffic, especially for low-latency traffic which requires the function deployment as close to the user as possible.

Based on the Central Office Re-architected as a Datacenter (CORD) platform [25], Mobile-CORD (M-CORD) [26] seeks to implement an SDN and NFV-based cellular network spanning both the RAN and CN. M-CORD is a composition of a virtualization platform, a virtual infrastructure management platform, an orchestration solution, and a network operating system (NOS). Network control is exercised through a distributed implementation of xRAN running as an application on the NOS, while network slicing is implemented via the ProgRAN application. In essence, M-CORD can be regarded as an extension of the xRAN, and, while it improves upon certain aspects by extending the control and user plane separation and bringing control functionality down to the RU level, it also suffers from similar drawbacks such as limited fronthaul adaptability, and limited use-case flexibility.

The NGMN Alliance has also put forth their version of the 5G architecture [27], wherein the use of a variety of RATs and functional splits is suggested for each type of traffic. While centralized control is implied, the architecture does not delve on the actual placement of control functions. The METIS-II project [13] follows up the ideas of the NGMN alliance by providing a highly-detailed RAN framework based on the CU-DU concept explained above, with support for several functional splits and multiple RATs. However, we note the absence of interface definitions between the control and data planes.

Similar to METIS-II, the 5G NORMA project [28] provides a detailed architecture with features such as multi-layer control, CU-DU functional splits, and support for multi-RAT. In the proposed architecture, the DU is RAT-specific, and therefore it cannot be re-used for

different RATs. On the one hand, it decreases the cost and provides better migration since a legacy transmitter device is used for each RAT (e.g., an eNB for LTE, an access point for Wi-Fi). On the other hand, an approach of this kind decreases flexibility. The architecture also includes frameworks for RRM and mobility management.

Furthermore, the X-HAUL project [29] supplements the ideas in 5G NORMA with backhaul and fronthaul solutions. X-HAUL proposes a hierarchical three-tier control plane– the L0 Controller which is responsible for a given area, the L1 Controller which exercises control over a set of L0 controllers and logically performs path setup across areas, and Top Controller which includes multiple L1 controllers in its domain.

To summarize, we note the following shortcomings in the prior work: (i) physical centralization of control functionalities, (ii) virtualization without consideration for SDN, (iii) static function distribution, (v) absence of radio resource virtualization solutions, and (iv) a lack of support for multiple RATs. While physical centralization decreases development and deployment costs, it also leads to poor scalability and flexibility. To this end, ARBAT has been designed with a view to overcome the aforementioned drawbacks.

2.2 CubeSat-Based IoT and Broadband Services

In this section, we survey the state-of-the-art in related activities worldwide, a summary of which has been provided in Table 2.1. Furthermore, we also discuss the existing SDN and NFV-based solutions for satellite networks. At the outset, we note that in addition to the solutions in Table 2.1 many enterprises such as AT&T, Samsung Research America, and Iridium Communications are working towards enhancing or providing service through satellite backhauls [30].

In the U.S., Iridium Communications Inc. has been replacing its original constellation with new Iridium NEXT satellites which support payloads of up to 50 kg, called SensorPODs, for Earth and space sensing in addition to housing communications infrastructure. The SensorPOD has a dimension of $20 \times 20 \times 14$ cm^3 and a weight of 4 kg [38]. However, it

Table 2.1: CubeSat-based IoT and broadband services

System	Astrocast [31]	Fleet [32]	Kepler [33]	Aistech [34]	Myriota [35]	Planet Dove [36]	Lacuna Space [37]
Purpose	IoT and M2M	IoT	Satellite backhauling	IoT, M2M, asset tracking	IoT	Earth imaging	IoT and M2M
ISL Capability	Yes	No	Yes	NA	NA	No	No
Orbital Altitude	575 km	580 km	600 km	500 km	800 km	420 km	500 km
Orbital Inclination	97.4°	NA	98.6°	97.8°	NA	52°	NA
Number of Satellites	64	100	140	102	50	150	32
Weight	8	20	7	NA	NA	NA	11
Operating Frequency	L-band ($1-2$ GHz)	NA	Ku- ($12-18$ GHz), and Ka-bands ($27-40$ GHz)	NA	UHF-band (400 MHz)	S-band ($2-4$ GHz)	ISM-band (2.4 GHz)
Form Factor	3U CubeSat	12U CubeSat	3U CubeSat	2U CubeSat	3U CubeSat	3U CubeSat	6U CubeSat

is not self-sustained and can only communicate with the host Iridium NEXT satellite, which means it has to forward the sensing data to its host satellite, and then the host Iridium NEXT satellite can either relay the data to its peers, or forward the data to receivers on Earth. Hence, there are no Inter-Satellite Links (ISLs) between SensorPODs, and the range of applications and services is very narrow. Late 2018 saw the Federal Communications Commission's approval for SpaceX's proposal to construct a broadband network named "Starlink" consisting of more than 12,000 satellites in total [39]. Among these satellites, 7,518 will operate at a 340 km altitude in the V-band ($40-75$ GHz), while 4,425 will orbit at a 1,200 km altitude, and operate in the Ka- ($26-40$ GHz) and Ku-bands ($12-18$ GHz). However, these satellites are not CubeSats, and are expectedly costlier to design and manufacture compared to 3U or 6U CubeSats [40].

Several other companies in countries including Switzerland, Australia, Canada, and Spain are also planning to launch their CubeSats to provide IoT and M2M communications services. Based out of Switzerland, the Astrocast platform [31] plans to launch a satellite system with a constellation of 64 CubeSats to provide fixed satellite services to serve users with satellite phone calls in fixed areas. However, in addition to using the already congested L-band, Astrocast allows for transmission of only 1 KB of data per day. Fleet in Australia

intends to use 12U CubeSats to connect to terrestrial IoT networks. But, like Astrocast, it is focused on providing low-bandwidth data services. Meanwhile, Kepler Communications in Canada plans to design and launch 140 KIPP 3U CubeSats with the intention of developing a satellite backhaul. However, 140 CubeSats are not sufficient to provide continuous coverage across all latitudes for an entire 24-hour period. Furthermore, apart from the use of capacity-limited 15 MHz channels for ground-satellite links, Kepler does not specify the inter-satellite communications capabilities of their constellation [41]. In Spain, Aistech intends to launch 100 6U CubeSats to support space imaging, and aviation and asset tracking services on Earth by 2022.

Within the realm of SDN and NFV-based solutions, we note the absence of a platform that targets CubeSats in particular. Instead, a bulk of the prior art is focused on extending SDN and NFV to LEO, MEO, and GEO satellites [42, 43, 44, 45, 46, 47, 48]. One of the first SDN-based architectures, OpenSAN [42], describes a possible deployment involving a GEO-based control plane, but does not delve into the implementational challenges such as those involved with maintaining a global network view across the SDN controllers. Sheng et al. [43] present resource management strategies for SDN-based satellite networks, along with the use of virtual network embedding, their approach however necessitates the need for a GEO relay. Under the VITAL project, Ferrus et al. [44] provide a detailed description of the use cases and benefits associated with SDN and NFV, however, the architecture they consider limits SDN/NFV to the terrestrial portion of the network only, with satellites merely providing a backhaul link. A similar approach is followed by Bertaux et al. [45] and Ahmed et al. [46], where the satellite operates on a bent-pipe principle relaying data from one terrestrial end-point to another. On the other hand, SERvICE [47, 49] takes a different approach by proposing an integrated space-terrestrial satellite communications network that delegates network management to ground stations, and network control to GEO satellites along with a multi-level data-forwarding scheme. However, the relaying of control information to LEOs via GEOs is a time-consuming affair, and, the increased control

traffic latency is undesirable.

In addition to these architecture frameworks, [50, 51, 52, 53] target different aspects of a software-defined satellite network, ranging from application specific implementations to protocol design. Of particular note is the Internet of Remote Things paradigm introduced in [54] and further explored in [55], wherein the authors utilize a satellite backhaul to provide connectivity to sensor fields in remote areas. However, the absence of SDN in [54] implies a lack of end-to-end control precluding service differentiation.

To this end, we note the lack of an integrated architecture that can provide differentiated services for a wide variety of applications in an end-to-end manner. Chief among the drawbacks noted in existing solutions are: (i) reliance on the bent-pipe paradigm, (ii) absence of end-to-end orchestration, and (iii) lack of consideration for advances in satellite infrastructure design.

CHAPTER 3
ARBAT: A FLEXIBLE NETWORK ARCHITECTURE FOR MODERN CELLULAR SYSTEMS

In this chapter, we introduce a variety of promising ideas for cellular systems and propose a new flexible wireless architecture called ARBAT.

3.1 Motivation

With the functional freeze for 3GPP Release 15 taking place in late 2018 [56], wireless networks have witnessed a sea-change over the past two years, with 5G being envisioned as a service delivery platform enabling a wide variety of use-cases, including but not limited to fixed wireless, enhanced Mobile Broadband (eMBB), massive Machine-type Communications (mMTC), and Ultra-Reliable Low-latency Communications (URLLC). This evolution of the cellular networking landscape has been driven, in no small part, by the ever-increasing demand for higher data rates resulting from the rise in popularity of bandwidth-intensive applications such as video conferencing, file sharing, social networking, and streaming services. Mobile data traffic has grown 18-fold over the past five years and is expected to exceed 49 exabytes per month by 2021 [57]. The highly differentiated nature of services that are expected to be deployed over or make use of cellular infrastructure in the near future all pose different requirements in terms of very high data rates and minimum latency that are difficult to achieve using existing cellular networks. More specifically, the monolithic nature and "one-size-fits-all" approach of 4G make it ill-suited to offer service-tailored connectivity with a wide variety of Quality of Experience (QoE) guarantees. Accordingly, the 5G cellular system requirements [58] mandate the following:

- peak data rates of up to $20\ \mathrm{Gbps}$,

- user experienced data rates of up to $100\ \mathrm{Mbps}$,
- three-fold increase in spectral efficiency compared to LTE,
- support for up to $10\ \mathrm{Mbps/m^2}$ area traffic capacity,
- user-plane latency of less than $1\ \mathrm{ms}$ in the RAN, and
- 100-fold increase in network energy efficiency.

To meet these divergent or even mutually exclusive technical requirements at a reasonable price, a 5G system shall have the following properties.

- **QoE-awareness:** The 5G system is not just a pipe for data transmission. In contrast, it is used to deliver certain services, each of which imposes Quality of Service (QoS) requirements. The 5G system shall understand the requirements for each particular application and serve its traffic accordingly.
- **Adaptability and Flexibility:** The 5G system shall suit a variety of use-cases right from eMBB to URLLC, and even perform use-case specific reconfiguration if required. In contrast to 4G systems with rigid Radio Access Technology (RAT) and inflexible infrastructure, the 5G system shall be able to adaptively reuse hardware and radio resources in order to satisfy ever-changing user demands.
- **Scalability:** The 5G system shall be able to support a large number of users with different needs. The radio, computational, fiber resources shall be used in a very efficient way. For that, the architecture shall support simultaneous use of various techniques that optimize spectral efficiency in different scenarios, e.g., simultaneous delivery of both massive data flows and light traffic. Moreover, the topology and the routes in the CN shall be changed dynamically to balance the load and avoid congestion.
- **Self-healing and Reliability:** In case of malfunction of any devices, including critical ones such as packet gateways, or controllers, the system shall be able to reassign their

functionality to other devices without any service degradation.

- **Multiple RATs and Backward Compatibility:** Due to the significant investment involved in infrastructure upgrades, the 5G system will not immediately replace other types of wireless networks. Additionally, there exist a wide variety of technologies such as Wi-Fi, Sigfox, and LoRa that cannot be beaten by 3GPP RATs at least in the near future. To provide compatibility with the devices supporting these technologies and to achieve higher efficiency of radio resource usage, the 5G system should integrate these RATs.

- **High Energy Efficiency:** In the cellular network, the RAN accounts for over half of the total power consumption [59]. However, the move towards a disaggregated RAN provides greater flexibility in resource allocation, and network planning and deployment. For example, architectural support for techniques such as joint optimal resource allocation that maximizes system throughput while minimizing energy consumption, configurable network density, and energy harvesting plays a vital role in lowering the carbon footprint of the network.

- **Low CAPEX and OPEX:** Generational upgrades have a major impact on a communications service provider's CAPEX [60]. Therefore, the appeal of any new architecture is also closely tied to its economic feasibility. In particular, those architectures that can demonstrate significant cost savings, when compared to legacy solutions, as well as the competition, over a period of $5-7$ years will see larger deployments. In order to minimize CAPEX, in contrast to previous generations of cellular systems with multiple overlapping networks of different operators, there would be a single 5G infrastructure which will share resources between Service Providers (SePs). Moreover, the architecture of the deployed 5G system shall allow the use of low-cost interchangeable and programmable devices, while the deployment of new services shall require no hardware upgrades.

To this end, we have previously identified 10 key enabling technologies that are essential for meeting the requirements put forth by 5G [61], chief among which are SDN and NFV.

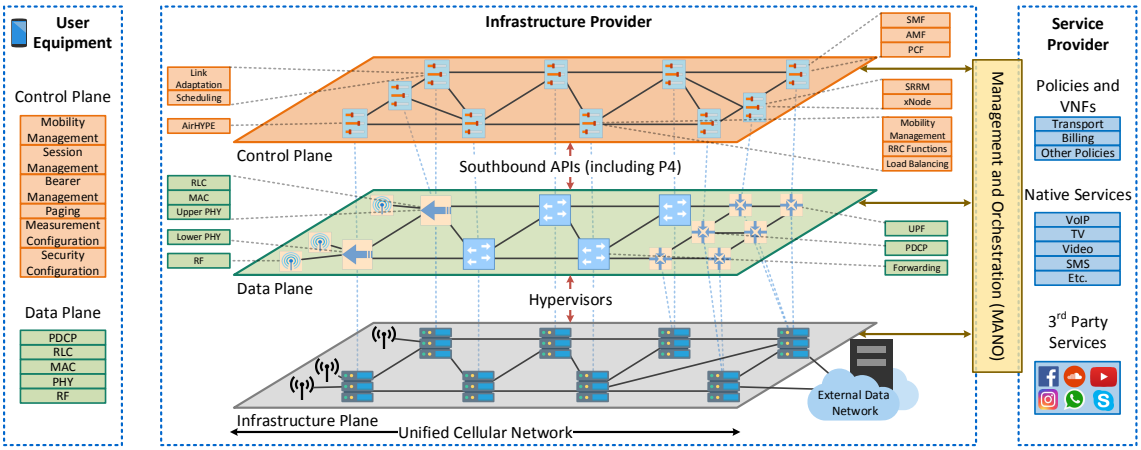


Figure 3.1: The ARBAT architecture.

In this chapter, we introduce the most promising ideas for 5G and beyond systems and propose a new flexible wireless architecture called ARBAT. More specifically, we introduce several innovative features such as the Universal Network Device (UND), the Unified Cellular Network (UCN) concept, multi-slice modular resource management with the AirHYPE wireless hypervisor, network–user application interaction through the xStream platform, and simplified multi-tenant orchestration through ServiceBRIDGE.

The rest of this chapter is organized as follows. We continue our discussion by presenting the ARBAT architecture and detailing the major components of the system in Section 3.2. Furthermore, we describe the novel features of the ARBAT infrastructure plane, data plane, control plane and management and orchestration (MANO) entity in Sections 3.3-3.6, respectively. We perform a qualitative evaluation of ARBAT by comparing it to other existing architectures in Section 3.7. Finally, we conclude this chapter in Section 3.8 by highlighting the major contributions.

3.2 System Architecture Overview

3.2.1 System Domains

ARBAT has been designed to meet and exceed the 5G system requirements identified in Section 3.1. Based on the concepts of SDN [62] and NFV [63], ARBAT consists of separate

data and control planes, an infrastructure plane and a MANO framework for Virtual Network Function (VNF) deployment and management, as shown in Figure 3.1. Below we briefly describe the primary components of ARBAT.

- **Infrastructure Plane:** It represents the physical and virtual hardware in the network. It consists of Wireless Transmission Points (WTPs) that use radio resources to transmit signals, network nodes that provide computing resources, interconnecting links that provide connectivity between nodes, and hypervisors that virtualize these resources. The infrastructure plane lies under the control of the Infrastructure Provider (InP), and infrastructure under a single InP forms an infrastructure domain. Through ARBAT, we introduce the concepts of UND (Universal Network Device) and UCN (Unified Cellular Network) in the infrastructure plane. Since ARBAT supports multiple InPs, there are multiple infrastructure domains in the network.

- **Data Plane:** Drawing on both CUPS and SDN, the data plane within ARBAT exclusively refers to the user traffic data path, i.e., the data plane does not contain any control functions, or carry control traffic. The data plane is characterized by the presence of network agents that serve as endpoints for control functions. Moving beyond OpenFlow, ARBAT makes use of P4 [64] to allow for custom data processing pipelines. While the network functions in the data plane belong to the SeP (Service Provider), their operation is governed through the control plane, by the InP with policy inputs from the SeP.

- **Control Plane:** It implements network control logic, through VNFs (Virtual Network Functions) called control functions. Common examples of control functions include scheduling, mobility management, and link adaptation. The control plane executes in-network control for network service operations and is managed by the InP.

- **MANO Framework:** It serves as the bridge between SePs and InPs, and is responsible for the overall orchestration and lifecycle management of both network services and network functions.

3.2.2 Stakeholders

With a view towards lowering infrastructure costs, we envision a single 5G infrastructure that will be utilized by several SePs. Recognizing that multi-tenancy is a core feature of next-generation wireless networks, the network should provide for a clear delineation of the purview of the different stakeholders associated with network operations– Infrastructure Providers (InPs) and SePs. The InP deploys and supports the infrastructure, provides pipes for SePs, and seeks to maximize resource utilization and operational profit, while the SeP seeks to maximize profit while operating under QoS requirements. More specifically, within ARBAT, where the InP and SeP are the major stakeholders, it is also necessary to outline the operational ownership of each plane. Following the concept of the light mobile virtual network operator (MVNO), ARBAT incorporates infrastructure, control and data planes that are entirely under the purview of the InP, and a MANO which is shared with the SeP. Such a separation is only one of many possibilities. Another approach could involve the InP having control over the infrastructure plane only, with the data and control planes belonging to the SeP.

Generally, SePs "provide" network services to end-users, while InPs "deliver" the contracted services to SePs. Examples of end-users include individuals or enterprises that require data or telephony services, while a cellular provider or an ISP are examples of SePs. An end-user is the SeP's customer, while the SeP is a customer of the InP itself. A SeP may contract with multiple InPs based on its requirements. More specifically, the SeP is responsible for providing the VNF models, VNF-Forwarding Graphs (VNF-FGs), specific policies such as those relating to restrictions on the placement of VNFs, preferred routes, etc., and Service Level Agreement (SLA) parameters such as throughput, latency, and reliability. The actual instantiation and lifecycle management of network functions and NFV Infrastructure (NFVI), network operations and control are all under the purview of the InP. Thus, the physical entities in the network are owned by the InP, while the logical entities are the SeP's property. Furthermore, we note that within the context of ARBAT, network

slicing [65] is an infrastructural construct, and not visible to the SeP. The InP may choose to make use of a slice for service delivery in case several services have common network functions. For example, if an InP has an active eMBB slice deployed over its infrastructure, and has to serve eMBB flows from several SePs, then these can be accommodated within a single eMBB slice. By separating the SeP from the day-to-day network operations, we can ensure that the InP has complete control over their infrastructure, and the need for resource isolation can be avoided, allowing for optimal resource utilization.

In the following, we describe in detail each of the three planes – the infrastructure plane in Section 3.3, the data plane in Section 3.4, the control plane in Section 3.5, along with the MANO framework in Section 3.6.

3.3 Infrastructure Plane

The infrastructure plane falls under the purview of the InP, with different InPs making their infrastructure assets available to the SePs for deployment of services. Physically, this plane consists of servers, switches, remote radio heads (RRHs), and interconnecting links, while logically the infrastructure can be modeled as a set of distributed resource groups, where each resource group is characterized by its physical location. The primary resources under consideration including computational, storage, memory, network, and radio resources. Such a multi-dimensional resource abstraction provides great flexibility to the SePs in the design of network services. To this end, we model the ARBAT infrastructure plane to consist of the following entities:

- **Universal Network Devices (UNDs):** This is the fundamental building block of ABRAT, and represents all the physical devices in the network, i.e., radio units, servers, switches, and legacy infrastructure.
- **Transport Links (TLs)**: They represent the physical links that connect different UNDs to each other.

In the following subsections, we describe the two entities in detail.

3.3.1 The Universal Network Device

The primary building block of ARBAT is the network node called the Universal Network Device (UND). The physical realization of the network consists of an interconnection between such UNDs forming a flat network that connects to other external data networks through private peering arrangements, or Internet exchange points. A UND can be a Physical Network Function (PNF) tied to specialized hardware, a Virtualizable Whitebox (VWB) on which VNFs are deployed, or a combination of the two. The UND is described by a set of hardware resources and built-in PNFs. Furthermore, we note that the introduction of the UND does not imply that the network consists of the same or similar devices. Rather, the UND can be viewed as a logical construct that helps in describing and parameterizing the huge variety of networking devices to achieve optimality in resource utilization, and flexibility in network design. Moreover, the UND provides the capability to extend multi-access edge computing (MEC) as close to the user as possible.

The PNF representation of a UND is motivated by two major factors. First, PNFs are used to represent legacy infrastructure that cannot be virtualized, e.g., LTE eNodeBs, legacy access points, non-programmable switches, etc. Thus, ARBAT can use existing LTE infrastructure, simplifying migration from old RATs to newer ones. Second, we note that the implementation of functionality associated with WTPs (Wireless Transmission Points) such as analog-to-digital/digital-to-analog conversion, up/down-conversions, and the duplexer function can be done far more efficiently, and at a lower cost, in hardware as opposed to software. Therefore, WTPs are also represented by the PNFs. This is especially important for many existing RATs, such as Wi-Fi, whose access points can be seamlessly integrated within the ARBAT architecture. In more general terms, WTPs within ARBAT are functionally equivalent to RRHs commonly found in existing architectures. Thus, the PNF representation allows ARBAT to function with and virtualize a variety of front-ends that are

readily available commercially.

The UNDs may also have additional computational resources, such as processing, memory, storage, etc., which are amenable to virtualization and support the deployment of VNFs. For example, a UND with the WTP PNF and additional computing resources may also execute all other data plane functions for a URLLC service in order to reduce latency. We can further differentiate between virtualizable hardware based on the presence of specialized features such as FPGAs, or support for the P4 language. More specifically, FPGAs are better suited for functions related to baseband processing than commodity CPUs, while P4 is a high-level language for programming protocol-independent packet processors. Unlike the OpenFlow specification that explicitly specifies protocol headers on which it operates, P4 suggests that switches should support flexible mechanisms for parsing packets and matching header fields, allowing control functions to leverage these capabilities through a common, open interface. In particular, ARBAT incorporates support for P4 to provide a low-latency data path for user traffic. The motivation for which comes from the fact that the functions performed by certain VNFs, such as those relating to packet forwarding, are not inherently suited to general purpose computing platforms. In other words, deploying such VNFs on a general purpose processor (GPP) will lead to a phenomenon similar to slow path processing [66] causing the network to experience widespread congestion.

However, this bottleneck could be alleviated if these functions were to be performed by dedicated switching hardware operating at line speed. Introduced in 2008, OpenFlow was a major step in providing access to line-rate forwarding, however OpenFlow presents a fixed-function data plane pipeline, which is difficult to extend and modify. By enabling custom pipelines that can be loaded and controlled on-demand, P4 presents a significant opportunity to realize such data forwarding related functions of the cellular network over high-speed switching hardware. For example, a majority of the functions performed by the User Plane Function (UPF) introduced by 3GPP [67] are amenable to a P4-based implementation, wherein the UPF control logic is implemented as a VNF on a GPP, while the packet

processing is done on the forwarding device.

3.3.2 Transport Links

The transport links (TLs) in ARBAT represent the physical interconnections between UNDs, and consist of both wired and wireless links. More specifically, we consider the following types of TLs.

- **Wired:** Fiber, coax, and copper.
- **Wireless:** sub-6 GHz, microwave, and mmWave.

For a more detailed overview of the different solutions that can be deployed on these links, we refer the reader to [68]. Additionally, in the context of ARBAT, fronthaul exclusively refers to the TL that terminates at a UND represented by a WTP PNF, and makes use of eCPRI split option E [68] which is suited for time-domain IQ sample transport. While the use of eCPRI imposes high throughput and low latency requirements on the fronthaul, we envision such PNFs being either co-located with, or deployed sufficiently close to VWB UNDs that implement the PHY layer functions, in addition to the use of IQ compression [69] where required. For other physical layer splits that may exist between UNDs, eCPRI options A through D are used, depending on the use case. For example, centralization of baseband processing functions at a single UND requires a lower physical layer split with high throughput and low latency demands, on the other hand, performing the baseband processing in a distributed manner on WVBs close to WTPs relaxes the link requirements. The former approach allows for greater centralization gain by allowing for centralized scheduling and resource management at the cost of robust TL requirements, while the latter allows for the use of non-ideal TLs at the cost of centralization gain. The trade-off ultimately depends on the nature of service that is being deployed and its QoS requirements.

3.3.3 Resource Virtualization and Abstraction

Thus far we have described the primary building blocks of the infrastructure plane, namely, UNDs and TLs. From a network deployment perspective, the infrastructure plane can be modeled as an undirected graph, where the UNDs represent the nodes, and the TLs represent the edges. Consequently, the need for resource virtualization necessitates the use of hypervisors. The virtualization of computing resources– processing, memory, and storage has been investigated a great deal with several virtualization solutions such as KVM [70], LXC [71], Xen [72], and Hyper-V [73] being readily available. Similarly, there exist network hypervisors that provide more than one networking context per physical networking device to allow for the provisioning of differentiated services– FlowVisor [74] and its extensions [75] that support OpenFlow; and the recently proposed HyPer4 [76] and HyperV [77] hypervisors for virtualizing P4-based UNDs. However, the virtualization of radio resources remains an ongoing challenge [78, 79]. To this end, ARBAT introduces a new wireless hypervisor– AirHYPE. Described in detail in Section 3.5, AirHYPE is a major step towards ensuring that: (i) each service or SeP is presented with a set of virtualized radio resources, and (ii) the InP is able to achieve optimal resource utilization through the use of statistical multiplexing while maximizing profits.

UNDs within ARBAT are characterized by an n-tuple resource abstraction, which represents for each UND: (i) a set of available hardware units and their characteristics, and (ii) programming logic executed on this hardware. As for hardware, the UND may contain the following units: (i) RF-front ends which are characterized by the operating frequency bands and maximum supported bandwidth, GPPs or FPGAs with given processing capabilities, (iii) memory and storage, (iv) ASICs which execute specific functions, etc. Based on the available hardware, a UND can also execute some specific PNFs which cannot be changed and/or software which can be changed by the InP. In the latter case, the InP can virtualize UND resources and use them to run some VNFs. Depending on VNF requirements (e.g., in terms of memory and processing load), the InP can select the appropriate UND. For

example, UNDs with FPGA are more suitable for executing baseband signal processing functions, while control functions such as scheduling can be executed on UND with a GPP. Additional details on how an InP maps VNFs to UNDs are given in Section 3.6. In a manner similar to UNDs, the properties of a transport link can be defined by its capacity, delay, and reliability.

3.3.4 Unified Cellular Network

In ARBAT, the role of each UND is determined by the set of active VNFs deployed on it. Therefore, the UND represents a different context for each network slice, and there is no predefined CU or DU. For example, a UND can run only PHY VNFs for an eMBB slice, and almost all VNFs for URLLC slice. Consequently, there is no broad distinction between the CN and RAN, and through ARBAT, we introduce the concept of the Unified Cellular Network (UCN)

More specifically, the support for UCN comes from the Protocol Data Unit (PDU) session concept introduced by 3GPP [80] and the generic UND model proposed herein. First, we recognize that a PDU session is a logical connection established between a UE and a data network, and that the data exchanged between the two entities is processed by a set of network functions that would traditionally be classified as belonging to either the CN or RAN. Second, we note that, by definition, it is possible to deploy any network function on the UND provided that the given UND can meet the function's resource requirements. In this manner, the PDU session connectivity can be realized as a network function chain [81], which originates at the UND with WTP functionality, traverses a set of UNDs and terminates at the UND that peers with the external data network. The peering UND also serves as the PDU session anchor, e.g., this could be the UND which hosts the UPF function [82].

The use of UCN within ARBAT provides certain key advantages:

- **Flat Architecture:** By eschewing the classical concepts of CN and RAN, UCN allows for the implementation of a flat architecture devoid of any hierarchy. A distributed flat

architecture serves to achieve much low user traffic latency than hierarchical solutions [83]. The reason for this is that in a hierarchical architecture, user traffic must be routed through the so-called CN nodes. By introducing an anchor, hierarchical cellular networks reduce the flexibility with which user traffic can be routed to the external data network, to the detriment of network latency. On the other hand, with UCN, ARBAT can set up service chains that are optimized for traffic delivery to data networks, without any topological restrictions except those relating to node and link capacity.

- **Network Deployment Flexibility:** UCN simplifies the mapping of network functions on to the infrastructure plane by not classifying UNDs on the basis of RAN and CN. Furthermore, UCN also brings flexibility to deployments, as the same UND may be used for both RAN and CN functions for different services, thus allowing for increased infrastructure sharing.

- **Reduced CAPEX and OPEX:** UCN is expected to lower costs for both the InP and the SeP. First, from the InP's perspective infrastructural constraints associated with only a subset of the hardware belonging to the core network are now taken care of. Second, for the SeP, UCN allows universal placement of network functions on UNDs, thus avoiding the potentially high costs associated with deploying functions on dedicated virtualized core network hardware.

3.4 Data Plane

The data plane is responsible for forwarding user data between end-points (e.g., between a UE and a remote server or between two UEs). In ARBAT, connectivity between the end-points is realized as a chain of network functions (both VNFs and PNFs) deployed at UNDs. As mentioned in Section 3.3, depending on the UND's capabilities, different network functions can be deployed on it. Let us describe how the data flow is forwarded from a UE to a remote server. The first UND on the path shall have an RF front-end in order

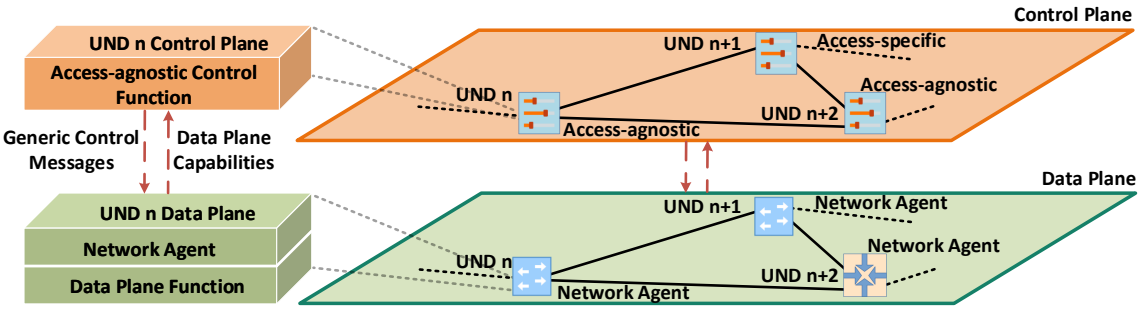


Figure 3.2: Control and data plane interaction via network agents.

to transmit/receive signals to/from UE over the wireless channel, i.e., this UND serves as the WTP. Furthermore, this UND and/or neighboring UND(s) shall execute RAT-specific network functions (e.g., for LTE RAT, these UNDs can execute baseband signal processing, MAC, RLC, and PDCP functions). After that, data packets are forwarded to other UNDs which perform forwarding functions all the way to the remote server.

As detailed in Section 3.6, for each service, the MANO framework configures and deploys a specific network function chain by taking into account available resources at UNDs and transport links, and service QoS requirements. In other words, ARBAT uses a dynamic functional split. For example, for an eMBB service, we can split RAN functions between several neighboring UNDs in order to implement CoMP and centralized scheduling solutions which allow increasing spectral efficiency. In contrast, for a URLLC service, all RAN functions must be deployed on a single UND which has an RF front-end because of tight latency requirements.

In multi-RAT scenarios, often it makes operational sense to have some access-agnostic control functions that issue generic commands over the SBI (Southbound Interface), in addition to access-specific control functions. On the other hand, the data plane functions are access-specific. Since the access-agnostic control functions must function with a variety of data plane functions, we introduce network agents. The access-agnostic control plane functions manage the data plane functions through network agents via the SBI, i.e., the network agents serve as the termination point for the SBI as shown in Figure 3.2. Thus, the

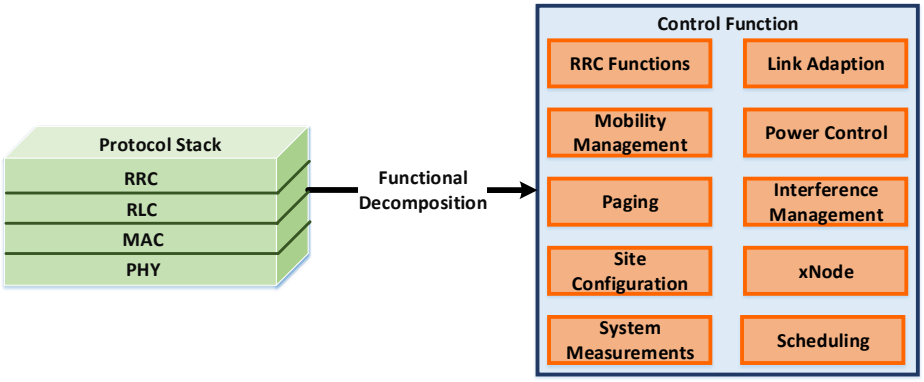


Figure 3.3: Functional decomposition of the cellular protocol stack.

agents are vital for native multi-RAT support, and the use of agents leads to a simplified SBI. They are also responsible for exposing data plane functionality to the control plane, for example, the PHY can expose PHICH, RACH, PRACH, PUCCH, and PUSCH parameters via a network agent to be configured by the controller, while the MAC data plane function could expose parameters such as MCS, PMI configuration, RB assignment bitmap, etc. Then, RAT-specific agents in the data plane convert generic messages to RAT-specific messages for the corresponding data plane functions. In this manner, the SBI is kept independent from the RAT in use, and any change in the supported RATs does not trigger a change in the SBI.

Taking into consideration a data forwarding function, the agent in question is the P4 agent which exposes the forwarding table to the corresponding control function. During the table update operation, the agent receives updates from the controller and modifies the match-and-action tables accordingly. On the other hand, if the underlying P4-capable hardware is to interface with an OpenFlow controller, the agent will perform the additional function of providing an OpenFlow-to-P4 mapping. Therefore, we see that the use of an agent allows the independent evolution of the data and control planes. In the aforementioned example, the same data plane function can interface with two different control plane functions.

3.5 Control Plane

It is responsible for network control and performance optimization. Following the SDN paradigm, the network functions can be categorized into control and data plane functions, where the decisions are made by the control plane functions, while the implementation of these decisions is carried out by the data plane functions. As shown in Figure 3.3, the non-exhaustive list of control plane functions includes:

- **Mobility Management:** Associate UE to a particular cell(s).
- **Paging:** Notify inactive UEs about incoming flows.
- **Scheduling:** Allocate resource blocks to different UEs/flows.
- **Link Adaptation:** Set the transmission scheme (modulation and coding scheme, number of MIMO layers, etc.)
- **Power Control:** Allocate power for each resource block.
- **System Measurements and Reporting:** Gather statistics (e.g., Channel State Information, buffer status) used by other functions.
- **xNode:** Provide interface between user-applications and the network as proposed in [84].

Based on the decisions made by the control plane functions, the data plane functions at different layers of the protocol stack process data packets. In other words, they add, remove, or modify headers, concatenate or segment data packets to create a transport block of a given size, encode or decode transport block(s), generate signals with the given power and modulation scheme, and finally, transmit or receive them over the wireless channel. As mentioned in Section 3.4, control functions can either be access-agnostic or access-specific. Access-agnostic control functions are not tied to a particular RAT and provide generic control functionality. For example, if the InP deploys multi-RAT joint scheduling over their

network, the scheduling function would be access-agnostic. xNode is another prominent example of a potential access-agnostic control function. On the other hand, access-specific control functions cater to specific RATs. Every control function can be designed to be access-specific, but the design of access-agnostic functions requires careful consideration of the system features that can be handled in a generic manner.

The use of NFV allows for the deployment of control and data plane functions at different UNDs. The placement of control functions shall take into account latency constraints. For example, the decisions made by scheduling and link adaptation functions must be provided to the corresponding data plane functions (e.g., MAC/PHY responsible for creating transport blocks) almost instantly. Such tight latency restrictions imply that these control functions shall be deployed close to the corresponding data plane functions.

The functional decomposition of the protocol stack for different RATs [56, 13] is a generally well-studied topic, and therefore we do not delve into it in this section. Instead, we will focus on the novel features of the control plane introduced in ARBAT. First, we introduce a novel control plane entity called xNode that enables communication between applications and the network. Thanks to this feature, we can easily classify traffic and obtain specific QoS requirements for each data flow. Second, we introduce a multi-slice RRM framework that utilizes information collected by xNode and distribute radio resources between various flows to satisfy their specific QoS requirements and optimize QoE for the end users.

3.5.1 xNode

To enable QoE-aware resource management, ARBAT incorporates the xStream platform introduced in [84]. This platform provides duplex communication between applications running on endpoint devices and the network. It is a flexible platform and can be used to improve performance for various types of traffic. In particular, in [84], we propose a set of solutions aimed at significantly increasing QoE metrics for web and video traffic.

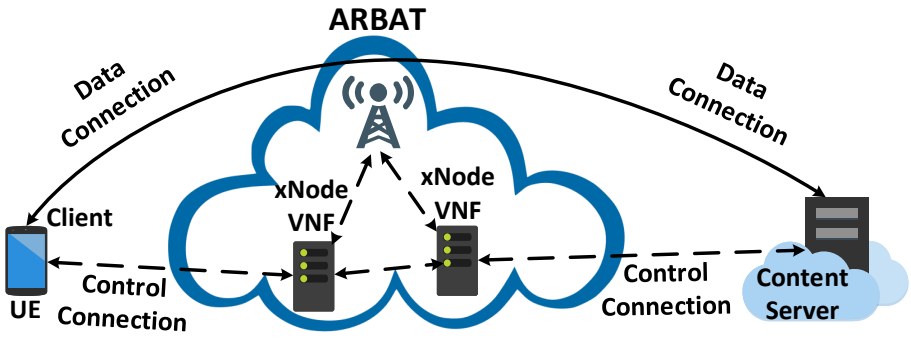


Figure 3.4: The xStream platform.

Following the idea of xStream, we introduce xNode, which is a network function providing an interface for control information exchange between the network and endpoint devices. Such communication is organized via a separate TCP connection which is established between an xNode and either a client application running on the UE or a server application in the Internet, as shown in Figure 3.4. In turn, xNode interacts with other control plane functions (e.g., functions responsible for resource allocation) to improve network performance. Note that multiple instances of xNode can be deployed in order to balance control traffic. Also, it is possible to use a separate xNode for each specific service. For example, in the case of URLLC service, xNode shall be deployed very close to the data path to provide low latency control information exchange.

Communication between applications and the network via xNode provides the following benefits. First, the application can directly inform the network about types of generated data flow (e.g., VoIP, Video, Web), enabling easy and accurate traffic classification without sophisticated deep packet inspection and/or machine learning algorithms used in existing networks [85]. Second, the application can provide the network with: (i) specific QoS requirements of a particular flow, (ii) forthcoming traffic characteristics, and (iii) the current state of the application. Given this information, the network can fairly allocate resources to maximize the overall performance, while in traditional architectures, the applications always compete for resources.

For example, the network can temporarily give more resources for a video client with

a small buffer to avoid video playback interruption, if the quality for other users does not suffer. Additionally, the network can inform the application about available resources and expected transmission characteristics, which can be used by the application to generate traffic accordingly. For example, adaptive video streaming applications can select an appropriate video bitrate/resolution by taking into account the expected link throughput signaled by the network.

3.5.2 Multi-Slice Radio Resource Management

Radio resource is the most expensive and scarce resource in a wireless system. Therefore, the fundamental problem is to design efficient radio resource management (RRM) algorithms. As mentioned in Section 3.1, in 5G systems, this problem is complicated by the need to share radio resources between various services with substantially different QoS requirements and also between various SePs. In the existing architectures, the network slicing typically uses isolated resources [86, 87]. The network is divided into slices, which serve traffic associated with a particular service (e.g., eMBB, URLLC), and each slice obtains a non-conflicting set of radio resources. With this approach, for each slice we can use a specific RAT and scheduling algorithm that takes into account slice QoS requirements. For example, for an eMBB slice, the scheduler can maximize throughput, while for a URLLC slice, the scheduler shall satisfy tight latency and reliability requirements consuming the minimal amount of radio resources. Moreover, such isolated slices are very favorable for SePs, since they can implement their own scheduling policies independently from InP. However, this approach degrades spectral efficiency and reduces overall performance. Since resource sets are isolated, one slice cannot use the resources allocated to another slice, even if those resources are not in use currently. Moreover, resource isolation diminishes channel diversity gain, i.e., UEs from different slices cannot use all resources to find resource blocks with the highest quality, which, in turn, reduces spectral efficiency.

The problems described above can be partially addressed by using a common multi-slice

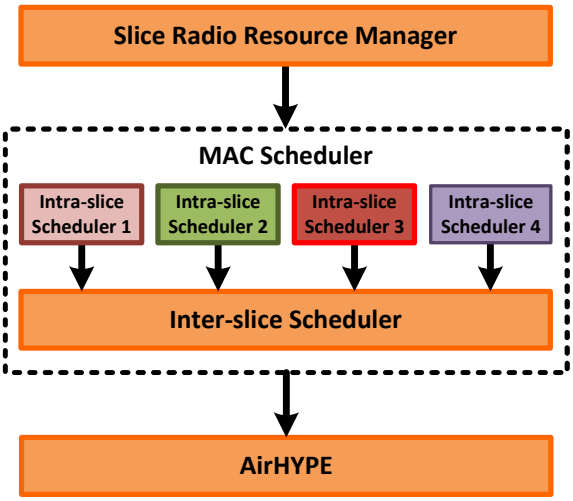


Figure 3.5: Multi-slice RRM framework.

scheduling algorithm which takes into account the QoS requirements for each slice. Such an algorithm would dynamically distribute resource blocks every TTI between slices and between flows in each slice. It would typically be RAT-specific and deployed at (or close to) the UND that performs MAC layer functions. However, in ARBAT, for various slices different functional splits can be used. In particular, to implement CoMP techniques, for the eMBB slice, the corresponding scheduler can be deployed at a UND controlling several UNDs with RF front-ends. On the contrary, for a URLLC slice the whole protocol stack and the scheduler shall be deployed at a UND with RF front-end because of tight latency requirements. Thus, we cannot use a joint scheduler for eMBB and URLLC slices in the considered scenario.

To enable flexible and efficient resource sharing between various slices, we propose a multi-slice RRM framework as shown in Figure 3.5 that utilizes virtualized radio resources, and consists of the following components: (i) Slice Radio Resource Manager (SRRM) for long-term resource allocation, (ii) multi-slice MAC scheduler for short-term allocations which consists of a set of intra-slice schedulers and an inter-slice scheduler, and (iii) a wireless hypervisor (AirHYPE) for resource virtualization and conflict resolution. In the following, we describe each of these components in more detail.

SRRM operates on a long-term timescale and determines the average amount of radio resources that should be allocated to each slice. For that, SRRM can use information obtained from applications via xNode (e.g., number of flows in each slice, their QoS requirements and characteristics). For each slice, SRRM determines: (i) appropriate RAT(s), and (ii) radio resources that can be used to serve this slice. Note that SRRM can allocate overlapping sets of virtualized radio resources for several slices. In this case, SRRM shall determine the long-term share of resources that can be used by each slice. The decisions made by SRRM and slice QoS requirements are signaled to the MAC scheduler.

The MAC scheduler operates on a short-term timescale (i.e., every TTI) and determines which resource blocks (RBs) and which transmission points (RF front-ends) shall be used to transmit a particular data flow. Since in our framework, slices can use an overlapping set of resources, we allocate RBs to flows belonging to different slices in two stages. At the first stage, for each slice, the intra-slice scheduler considers all RBs and pre-allocates them to the corresponding flows using a slice-specific policy. The policy can take into account slice QoS requirements, and, moreover, it can be provided/configured by the SeP (e.g., SeP can prioritize some specific flows). Since slices can use shared resources, the same RBs can be allocated to flows belonging to different slices. Thus, at the second stage, the inter-slice scheduler resolves such a conflict and selects a single flow to serve in the given RB. For that, the inter-slice scheduler can use the following information: spectral efficiency of competing flows in the considered RB, the average share of resources allocated to the corresponding slices, QoS/QoE requirements of the flows, etc. The actual policy used by the inter-slice scheduler is determined by the InP, but this policy shall ensure that in the long-term each slice obtains a share of resources given by the SRRM. An example of a multi-slice MAC scheduler can be found in [84]. In the section, we consider a scenario with web and video slices and design intra- and inter-slice scheduling policies which improve QoE for both types of traffic. In the proposed RRM framework, MAC schedulers can be deployed at different UNDs and can use overlapping sets of resources. In particular, this is needed to

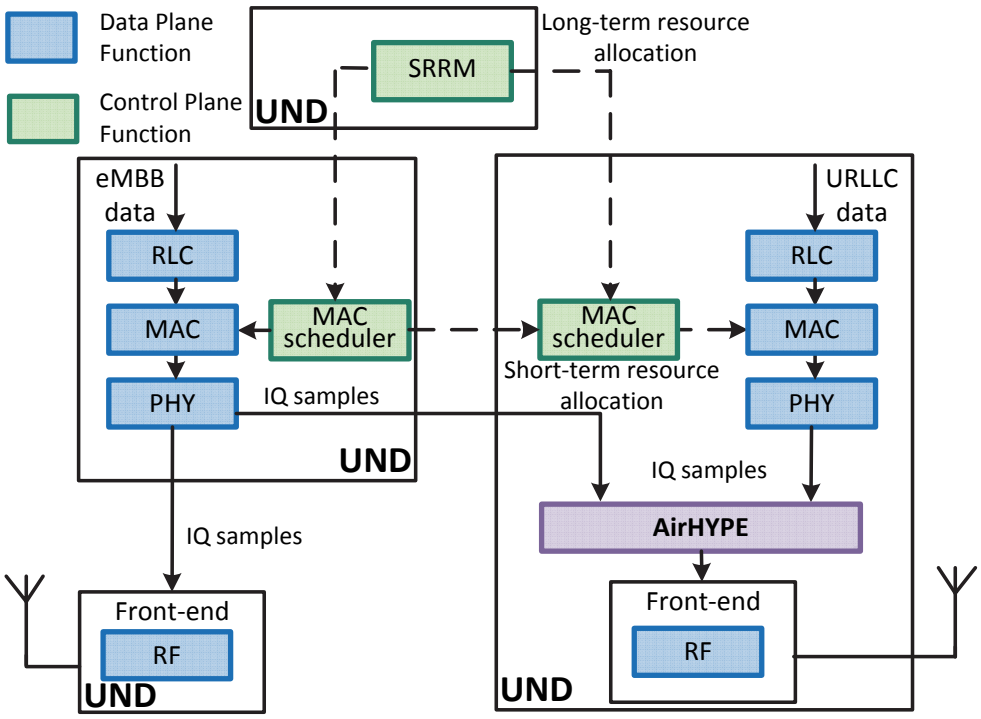


Figure 3.6: Operation of multi-slice RRM framework in case of eMBB and URLLC resource slicing.

enable efficient resource sharing between eMBB and URLLC slices as shown in Figure 3.6. In the considered scenario, due to tight latency constrains all functions for a URLLC slice (including MAC scheduler) shall be deployed at the UND with RF front-end (see right part of Figure 3.6). In contrast, for an eMBB slice we can deploy the MAC scheduler at the UND controlling several RF front-ends (e.g., in order to implement CoMP technique). Thus, in the considered scenario, different slices (and their MAC schedulers) can use overlapping time-frequency resources that ultimately map to the same physical RF front-end. To resolve such conflicts and multiplex different data streams, we introduce a wireless hypervisor called AirHYPE.

Below we provide an example no how to design AirHYPE for OFDM-based RATs (e.g., LTE, Wi-Fi, and NR). First, to provide flexibility and RAT-agnostic operation, we recognize that AirHYPE must be placed as low in the protocol stack as possible. In particular, by placing AirHYPE between the lower physical layer and RF front-end, we can ensure that it

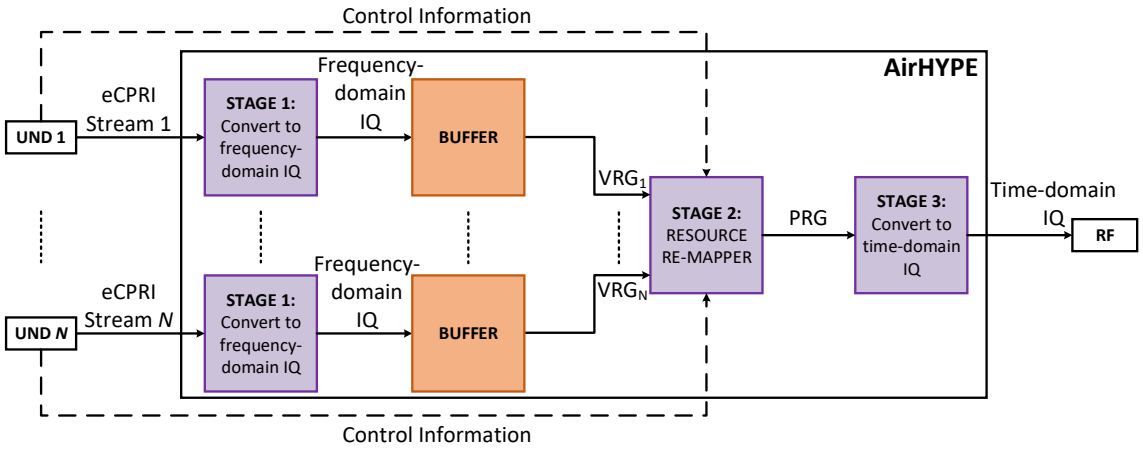


Figure 3.7: The AirHYPE wireless hypervisor.

processes only time- or frequency-domain IQ samples from different streams, the nature of which does not depend on the considered RAT. At the same time, we note that in its current form, ARBAT virtualizes radio resources for each RAT separately, and therefore AirHYPE does not multiplex streams across different RATs. Second, we design AirHYPE to make use of the existing signaling framework provided by eCPRI, thus eschewing the need for extra control overhead. The goal of AirHYPE is to multiplex several eCPRI streams into a single eCPRI stream, which is then passed to physical RF front-end.

The operation of AirHYPE is illustrated in Figure 3.7. First, multiple eCPRI streams $(1, \ldots, N)$ from different UNDs provide the input data, which consists of serialized time- or frequency-domain IQ samples. The first stage of AirHYPE is to convert each input stream to a frequency-domain IQ data set, where the i^{th} set represents the virtual resource grid (VRG) corresponding to the i^{th} input stream, with each VRG being stored in a buffer. We note that if the incoming data stream already consists of frequency-domain samples, then the first stage can be skipped. The second stage is a resource re-mapper which shall multiplex multiple VRGs into a single physical resource grid (PRG). The InP can deploy any multiplexing algorithm at the resource re-mapper. For example, as shown in Figure 3.8, if a URLLC stream is competing for a subset of the total physical resources required by an eMBB stream, then the re-mapper, by recognizing the higher priority of the URLLC

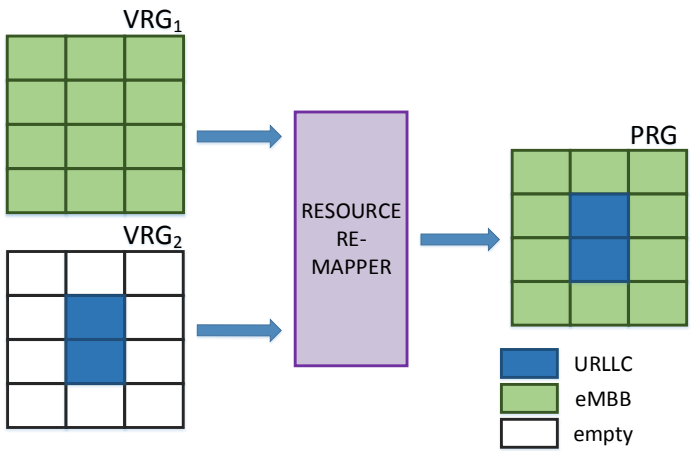


Figure 3.8: Resource re-mapper example.

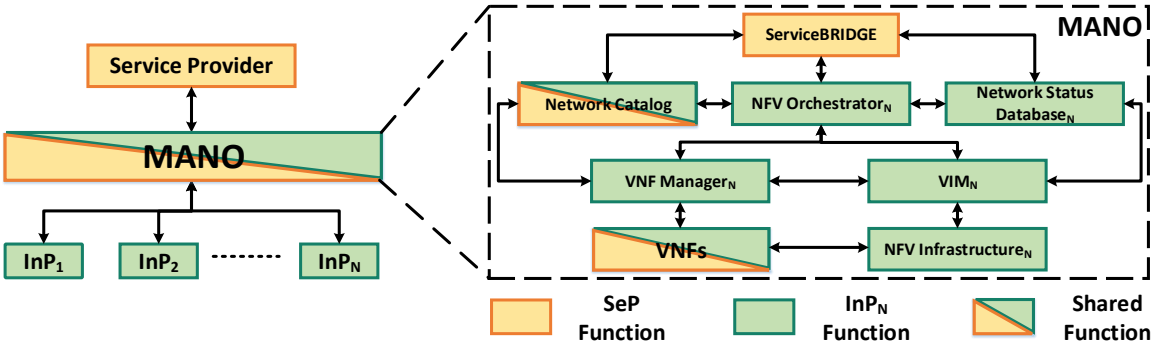


Figure 3.9: The MANO framework within ARBAT.

stream (the priority is provided with the control information within the eCPRI stream), can preempt the eMBB transmission and schedule the URLLC transmission. Another option is that the re-mapper can superpose IQ samples from different streams. In this case, the receiver can use successive interference cancellation techniques to demultiplex the original streams. Finally, at the third stage, the obtained PRG is converted to a time-domain data stream which is passed on to the RF front-end for transmission.

We should note that AirHYPE takes advantage of the bit synchronous interface of the eCPRI link and does not require any additional synchronization. Furthermore, it does not require any modification to the physical layer and follows a plug-and-play approach.

3.6 Management and Orchestration

The use of NFV within ARBAT requires a robust, yet low complexity, MANO framework as shown in Figure 3.9. The MANO framework is shared between the InP and SeP, and is primarily tasked with the management of virtualized infrastructure, orchestration of network services, and the lifecycle management of VNFs [88]. It typically consists of three major functional blocks: the Network Function Virtualization Orchestrator (NFVO), Virtual Network Function Manager (VNFM), and the Virtualized Infrastructure Manager (VIM), along with a Network Catalog for network service and network function definitions, that are commonly found in solutions such as the Open Network Automation Platform (ONAP) [89], and Open Source MANO (OSM) [90]. In addition, we introduce two new components– the Network Status Database (NSD) for monitoring the current network state, and ServiceBRIDGE that acts as an interface between the SeP and InP domains. Figure 3.9 also shows the interactions between the different functional blocks. Below, we provide detailed information about each of them.

3.6.1 Network Function Virtualization Orchestrator

As part of the MANO, the NFVO plays a key role in the system performance, overseeing the global network resources and allocating resources between network slices. The NFVO directly interacts with SePs and is responsible for fulfilment of SePs' requests for service. The NFVO receives from each service i, the set of network requirements R_i such as maximal packet delay, memory, capacity, etc. If the InP cannot fulfil all requirements for service, it is forced to pay penalty P_i given by SeP which may be monetary. Using the information about the current network state from the NSD, and templates from the catalog, the NFVO decides how to construct a network service and dynamically modify it, fulfilling requirements R_i for each service, and minimizing the total penalty $P_{tot} = \Sigma P_i$. The NFVO manages the lifecycle of each network service, including instantiation, scale-out/in, performance measurements,

event correlation, and termination.

3.6.2 Virtual Network Function Manager

The VNFM receives the corresponding set of requirements for VNFs from the NFVO. The VNFM is responsible for the lifecycle management of each VNF instance– scaling, changing operations, adding new resources, and communicating between states of VNFs and other functional blocks.

3.6.3 Virtualized Infrastructure Manager

Operating under the purview of the InP, the VIM is responsible for controlling and managing the Infrastructure Plane resources– computing, network, and radio; and therefore works in close cooperation with the computing, network, and wireless hypervisors. The mapping of physical resources to virtual entities, and the associated lifecycle management fall under the purview of the VIM. It is through the VIM's northbound interface (NBI) that physical and virtualized resources are made available to the VNFM and the NFVO. Furthermore, the VIM also organizes virtual links, networks, and ports, and is responsible for the management of the NSD repository described in the following section. In ARBAT, we envision multi-VIM, multi-site deployments that allow SePs to make use of infrastructure from multiple InPs.

It is worth noting that such an approach allows implementing dynamical multi-layer functional splits within the network. A static split poses fixed requirements regarding throughput and latency, which may be difficult to meet with a less-than-ideal fronthaul. Instead, a dynamic functional split adapts to fronthaul availability and use-case requirements. A dynamic functional split plays a major role in enabling use-case adaptability. Since we can map VNFs on different UNDs using the VIM in ARBAT, dynamic functional splits can be implemented seamlessly.

3.6.4 Network Catalog

The Network Catalog is a repository of available PNFs, VNFs, and network services that can be used by other blocks such as the NFVO and VNFM for service instantiation. The Network Catalog is a shared entity with its contents being owned by both the SeP and the InP. Each PNF and VNF in the catalog is represented by a template called the PNF Descriptor (PNFD) and VNF Descriptor (VNFD) respectively, which captures its deployment and operational behavior. On the one hand, a PNFD describes the functionality of a PNF, along with available interfaces. On the other hand, a VNFD contains information relating to the sub-components and their dependencies and interconnections, resource allocation criteria, and geo-location placement of a VNF. Similarly, network service behavior is captured by the Network Service Descriptor (NSD). The NSD consists of PNFDs and VNFDs associated with the constituent PNFs and VNFs, the VNF-Forwarding Graphs (VNF-FGs) that form the service, VNF dependency requirements, and interconnection link requirements. While the InP makes use of the Network Catalog to carry out deployments, the SeP is responsible for the onboarding of its PNFs, VNFs, and network services.

3.6.5 Network Status Database

The NSD holds information about the devices in the network, the links between them, and the deployed services. The NSD provides real-time information about resource usage and availability, along with service status. Given the potentially large and distributed nature of the NSD, it is implemented in the form of a NoSQL database, with the document store, key-value store and, graph databases being potential candidates for implementation. More specifically, the NSD contains the following information:

- **UNDs:** UID, special features (P4, FPGA, etc.), geographical location, resource availability, resource occupancy, and cost per unit of resource.

- **TLs**: Endpoints, type (wired or wireless), resource availability, resource occupancy, la-

tency, and cost per unit of resource.

- **Services:** SID, owner (tenant ID), constituent VNFs (VNF IDs), UNDs (UIDs), logical links (endpoints), resource utilization, and SLA (latency and throughput requirements).

Furthermore, all information stored in the NSD except for that relating to services is exposed to the SeP through ServiceBRDIGE, i.e., the SeP can only view information pertaining to resource availability and pricing.

3.6.6 ServiceBRIDGE

Given the strict separation of InP and SeP domains in ARBAT, ServiceBRIDGE is intended as an interface between these two stakeholders. More specifically, in ARBAT, the infrastructure and lifecycle management of network functions are the responsibility of the InP in order to achieve optimal resource utilization. For example, the InP is free to place VNFs on UNDs under its purview, so long as the SLA requirements from the SeP are met. However, this poses a challenge in a multi-InP environment, where the SeP may require a service that spans multiple InP domains. Since it is impractical to expect different InPs to converge to a common provisioning decision, we introduce ServiceBRIDGE as shown in Figure 3.9.

ServiceBRIDGE interfaces with the Network Catalog from the SeP, and with the NSD and NFVO from each InP. From the Network Catalog it obtains the VNF-FG associated with the service, and from the NSD it receives information about the resource availability in each InP's domain, along with the per-unit resource cost. The primary function of ServiceBRIDGE is to partition a large service chain covering multiple InP domains, into multiple small sub-chains that cover a single InP domain. Each of these sub-chains has their own set of SLAs derived from the SLA of the parent service chain. These sub-chains are delivered to the corresponding NFVOs, which then perform service orchestration based on additional information from the Network Catalog. The metric used for service chain partitioning is left for the SeP to decide. For example, the SeP could choose profit maximization as the basis

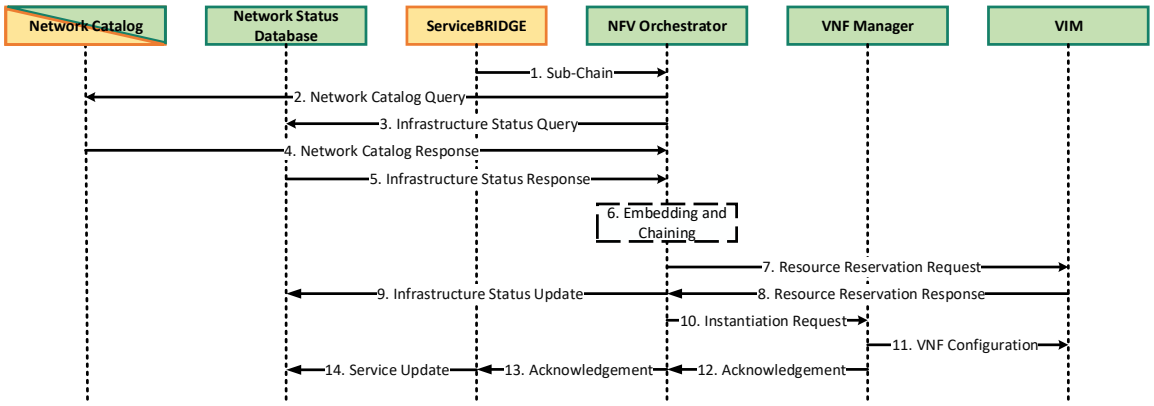


Figure 3.10: The service instantiation and delivery procedure.

for partitioning, or could choose to partition for lower operating costs by allowing flexibility in the SLA requirements of the sub-chains. On the other hand, since the chain partitioning takes place at a level above the NFVO, the InP is not aware of this procedure. From the InP's perspective, the NFVO component receives multiple sub-chains from different SePs simultaneously, and interacts with the Network Catalog, VNFM, and VIM to embed these chains on to the underlying infrastructure in a manner that achieves optimal resource utilization. Thus, the InP is only responsible for delivering the SLA associated with the sub-chain. To summarize, ServiceBRIDGE allows for provisioning of network services across multiple infrastructure domains, without the complexity associated with inter-domain interaction.

Within ARBAT, the Service Instantiation and Delivery procedure is responsible for provisioning network service requests from the SeP. In Figure 3.10, it is assumed that the service chain partitioning procedure has been completed by ServiceBRDIGE. In Step 1, ServiceBRIDGE delivers the sub-chain to the NFVO, which in turn requests the Network Catalog for details about the constituent VNFs, and the NSD for current network status, in Steps 2–5 respectively. Using the information received, the NFVO determines the optimal placement of VNFs over the virtualized infrastructure, and their interconnections as shown in Step 6. Step 7 involves the NFVO sending a resource reservation request to the VIM. Once the VIM performs the resource allocation, it sends an acknowledgment back to the NFVO, which in turn updates the NSD in Steps 8–9. Then, an instantiation request is sent

to the VNFM in Step 10, along with the information relating to instantiation parameters such as the resource requirements of VNFs, interconnecting link bandwidth requirements, scalability parameters, etc. The VNFM interacts with the VIM in Step 11, and configures the VNFs along with the interconnecting links over the virtualized infrastructure. Steps 12–13 represent the successive acknowledgements that are sent from the VNFM to the NFVO, and from the NFVO to ServiceBRIDGE. Finally, in Step 14, ServiceBRIDGE updates the NSD to reflect the newly provisioned service, and service delivery is complete.

3.7 Qualitative Evaluation

In this section, we perform a qualitative comparison of ARBAT with the state of the art SDMN solutions described in Section 2.1. Our comparison is based on the following properties.

- **Infrastructure:** The SDMN solutions we discussed in Section 2.1 employ SDN, NFV or a combination of both. While neither is necessary for the other to exist, a combination of the two allows for the implementation of a wide variety of features– policy-based control, network slicing, network automation, etc. and is the preferred solution. ARBAT implements both SDN and NFV at the grassroots level by introducing the concept of UND, which supports PNFs, and allows for resource virtualization that supports the deployment of VNFs. The system can consist of either monolithic or disaggregated base stations. Furthermore, the disaggregation can either be partial or full. In partial disaggregation only a limited part of the base station functionality is disaggregated, while in full disaggregation, there is complete flexibility in the distribution of base station functions. From a scalability and cost standpoint, a fully disaggregated base station, as used in ARBAT, would be the preferred option. At the same time, instead of using a hierarchical CN–RAN, ARBAT uses a UCN which provides design flexibility, and simplifies reconfiguration of the network and the deployment of new services.

- **Control Plane:** There are two aspects to control plane design. First, the control plane can be either be physically centralized or physically distributed, with the latter being preferred from a system scalability perspective. Second, the control plane function distribution can either be static or dynamic. While a static function distribution is easy to implement, a dynamic distribution allows the system to adapt to a variety of use-cases. ARBAT has a physically distributed control plane with dynamic distribution of control functions, which makes the 5G system scalable and flexible. Furthermore, ARBAT also introduces a novel user application–network interface, namely xNode, which significantly enhances traffic engineering.

- **Scalability:** Scalability is measured by how well the network responds to an increase in traffic, and is characterized by the absence of links or network components that are prone to congestion. A major benefit of ARBAT in this regard is that it scales well in response to an increase in the number of users, and the amount of traffic, before requiring provisioning of new hardware and additional capacity. The disaggregated approach to network design, decentralization of control functionalities, and UCN– all play a major role in enhancing the scalability. In addition, the use of UNDs adds an additional layer of reliability to the network, as network functions can be migrated easily to the nearest UND in the case of device failure.

- **Modularity:** There are two aspects to network modularity– hardware and software. First, from the hardware perspective, the addition of new components must follow a plug-and-play approach. To this end, the concepts of UND and UCN, that allow for rapid provisioning of resources and practically unlimited flexibility in network design, are two significant enablers towards a modular network. Second, from the software perspective, the set of VNFs deployed across UNDs can be modified easily using the MANO framework as has been described in Section 3.6. Thus, ARBAT exhibits significant modularity, and can seamlessly adapt to changing use-cases.

- **Fronthaul Adaptability:** The system should have provision for functioning with a variety of use-cases, including eMBB, mMTC and URLLC. To optimize spectral efficiency and to satisfy strict requirements for different slices, different functional splits are needed. In other words, use-case adaptability is tied to a dynamic and distributed control plane that allows changes in network function distribution, which also allows the architecture to function with a variety of fronthaul options. ARBAT supports flexible functional splits, which results in high fronthaul adaptability.

- **Multi-RAT:** In contrast to many other existing architectures, ARBAT is RAT-agnostic and supports multiple RATs via the same generalized architecture. Owing to the use of UNDs and network agents, ARBAT can easily integrate devices supporting existing and emerging RATs.

- **Network Slicing:** The use of network slicing allows operators to offer differentiated services over the same infrastructure while optimizing network resource utilization, and as such has been recognized by 3GPP as a key feature [15]. Therefore, support for slicing is a key required feature. While many existing architectures leave slicing out of consideration, or propose semi-static isolated slicing, ARBAT is the first-ever architecture with native support for dynamic slicing. The latter is especially important for RRM, since typically radio resources become a bottleneck for the entire system. The designed modular multi-slice RRM framework allows achieving high spectral efficiency and the fulfillment of QoS requirements for different slices.

- **Special MANO features:** To manage the virtualized infrastructure and orchestrate network services, ARBAT uses a rich MANO framework. In ARBAT, MANO is complicated because of the necessity to support the strict separation of InP and SeP domains. To address this issue, the proposed architecture uses ServiceBRIDGE as an interface between these two stakeholders.

To this end, we have summarized the main features of ARBAT in comparison with

Table 3.1: Feature comparison of ARBAT with key existing SDMN architectures

(a) Infrastructure, Control Plane, Scalability, and Modularity

System Architecture	Infrastructure	Control Plane	Scalability	Modularity
TIP vRAN [20]	Base station: Partially disaggregated CN–RAN: Hierarchical	No control & data plane separation	Limited (absence of SDN)	Limited (can add RRUs only)
NEC NFV C-RAN [21]	Base station: Fully disaggregated CN–RAN: Hierarchical	No control & data plane separation	Limited (absence of SDN)	Limited (can add RUs with L2 functions)
METIS-II [91]	Base station: Fully disaggregated CN–RAN: Hierarchical	Dynamic distribution of control functions	High	High (Fully modular RAN, not focused on CN)
O-RAN [92]	Base station: Partially disaggregated CN–RAN: Hierarchical	Dynamic distribution of control functions	High	Mixed (Fully modular CN and partially modular RAN)
ARBAT	Base station: Fully disaggregated CN–RAN: Unified	Dynamic distribution of control functions	High	High (UNDs and UCN)

(b) Fronthaul Adaptability, Multi-RAT Support, Network Slicing, and Special MANO Features

System Architecture	Fronthaul Adaptability	Multi-RAT	Network Slicing	Special MANO Features
TIP vRAN [20]	Limited (PHY layer split only)	Not Supported	End-to-End Network Slicing (E2E-NS)	No specific solution
NEC NFV C-RAN [21]	High (flexible functional splits)	Not Supported	No specific solution	No specific solution
METIS-II [91]	High (flexible functional splits)	Supported (RAT-specific and RAT-agnostic functions)	AIV-agnostic slicing	Spectrum Assignment Coordination
O-RAN [92]	Limited (single functional split)	Limited Support (LTE and Wi-Fi)	Through control applications	No specific solution
ARBAT	High (flexible functional splits)	Supported (UNDs and network agents)	Dynamic	ServiceBRIDGE

other system architectures in Table 3.1. Qualitatively, we consider ARBAT to be the most feature-complete SDMN architecture based on its infrastructure and control plane design, multi-layer RRM framework, ability to adapt to a variety of use-cases and fronthaul options, high scalibility, multi-RAT support and modularity.

3.8 Highlights

In this chapter, we have introduced a novel architecture for 5G and beyond wireless systems called ARBAT. ARBAT has many innovative features aimed at providing highly efficient QoE-aware communications in heterogeneous environments with low CAPEX and OPEX. More specifically, by following the virtualization paradigm and replacing the hierarchical

CN and RAN with a UCN consisting of UNDs, ARBAT is able to bring the external data network closer to users, thereby reducing latency and enabling the URLLC use case. At the same time, for highly efficient spectrum usage and operation with massive antennas, control functions can be deployed at a fewer number of central UNDs with higher computational capabilities. Moreover, the concept of UNDs allows ARBAT to integrate both legacy devices with hardware-defined PNFs, as well as make use of infrastructure resources for running various network functions efficiently.

Additionally, communication between applications and the network through the xStream platform combined with the novel multi-slice modular resource management framework aims at maximizing network capacity with respect to the provided QoE for different slices. On the one hand, xStream synchronizes network capabilities and application demands, while on the other hand, the non-isolated slicing paradigm avoids wastage of channel resources and allows for maximizing the user-perceived spectral efficiency, Furthermore, the modular resource allocation framework allows for MNVO-defined radio-resource schedulers for specific slices. The latter together with enhanced MANO augmented with ServiceBRIDGE greatly simplifies multi-tenant orchestration.

CHAPTER 4
RADIO ACCESS NETWORK DESIGN WITH SOFTWARE-DEFINED MOBILITY MANAGEMENT

Building upon the cellular network architecture introduced in Chapter 3, this chapter of the thesis focuses on the radio access network (RAN). In particular, in this chapter we introduce a new RAN design framework augmented with software-defined mobility management (SD-MM). The primary aim of the proposed framework is to assist with the transition from monolithic base stations within 4G to distributed network elements that are becoming increasingly common within 5G and beyond systems.

4.1 Motivation

As we have seen in the previous chapter, 5G and beyond networks are characterized by distributed network elements such as distributed units (DUs) and central units (CUs). However, a majority of the networks operational today are characterized by monolithic base stations that have been a staple of 4G and its predecessors. As network operators transition from 4G to 5G and beyond networks, the RAN design problem has become increasingly important.

In this chapter, we explore the RAN design problem from a mobility management perspective. At the outset, it is apparent that there are several inter-related problems to be solved for ensuring system operation with minimal overhead. For example, we must determine the optimal number of CUs required to serve a given number of DUs and users, and the placements of these CUs relative to the DUs. We also need to determine the association of each DU with the CUs that we have obtained in the previous step. Furthermore, the costs associated with registration and paging events need to be documented, and user-specific CU clusters generated for each user with the aim of minimizing the aforementioned costs. Finally, we note that the proposed framework specifically targets the RAN design problem,

and thus aims to be compatible with existing core network deployments.

To this end, the work presented in this chapter aims to address the problem of optimal CU planning and user-specific clustering against the backdrop of software-defined mobility management (SD-MM). Given the maximum number of users (UEs), DUs, a core network defined in terms of switches, links and controllers, and mobility metrics (call arrival rates and user mobility rates), we provide the following major contributions:

- A novel SD-MM framework designed to reduce signaling overhead.
- Accurate analytical characterization of traffic and delay-sensitive registration and paging costs.
- An optimization framework that determines the optimal number of CUs, DU-CU and CU-controller assignments, along with user-specific CU clusters for signaling reduction.
- Detailed performance comparison with conventional LTE/NR networks in terms of signaling costs.

To the best of our knowledge, this work is the first to address the CU planning and clustering problem, within the domain of software-defined next-generation cellular networks by taking into consideration mobility management procedures. In addition, unique to our solution is the fact that, we do not rely on fixed cost definitions. Deploying the RAN within a pre-existing core network allows us to perform accurate traffic characterizations and delay assessments over links that form the network. In turn, by using such dynamic costs, we seek to provide realistic formulations that match real world expectations. Performance evaluation is based on a cost comparison for different scenarios and changing network dynamics, with conventional LTE/NR networks that do not implement clustering. Our results show that an improvement of close to 80% can be obtained over existing systems, and that of around 70% when switching from single path routing to multi-path routing.

The architecture under consideration in this chapter consists of a software-defined radio access network (SD-RAN) in which each base station consists of a number of hardware-only

DUs and a software-based CU. The fronthaul link between a CU and its corresponding DUs is realized using fiber optic links. Such a decoupling is in line with the software-defined networking (SDN) paradigm and allows for independent evolution of hardware and software at each end. Furthermore, it allows the network structure to take a more dynamic form where DU-CU associations are not set in stone, but can be varied on-demand. The functional split between the DUs and their corresponding CU sees partial baseband processing being performed at the DU, while other higher PHY and MAC layer functionalities are implemented at the CU. This fine-grained base station decomposition allows for reduced stress on the fronthaul network and superior scalability. The use of network function virtualization (NFV) ensures that CU functionalities are implemented on commodity hardware, allowing for seamless migration from one physical site to another, as required.

The rest of this chapter is organized as follows. In Section 4.2, we describe the existing work in this domain. In Section 4.3, we provide the system architecture and present the mobility management framework. Then, Section 4.4 describes the optimal CU planning and clustering problem. In Section 4.5, we evaluate the performance of our proposed scheme, and Section 4.6 provides the key highlights of this research objective.

4.2 Related Work

At the outset, a majority of the prior art [93, 94, 95, 96, 97, 98, 99, 100] on SD-MM does not take the structure and design of the network into consideration. More specifically, the authors in [93] present a distributed mobility management framework that can either utilize PMIPv6, SDN, or a combination of BGP and DNS. While we note that the authors have provided a real-world implementation of their framework, the proposed framework is not geared towards cellular networks, and therefore limited in its applicability.

On the other hand, the work presented in [94] makes exclusive use of SDN and NFV for traffic, resource, and mobility management, along with an emphasis on throughput during handovers. Furthermore, the cost incurred by the system has been defined in terms of the

power consumption of component switches in the core network. While power consumption metrics help quantify the network's energy efficiency, in the absence of signaling-related performance benchmarks, it is difficult to judge the efficacy of the proposed framework. In [95], the concept of SD-MM is extended further to incorporate both signaling overheads and delays, with the system being deployed on a pre-existing RAN. The authors have also provided a Mininet based implementation of their framework.

In a similar vein, [96] also presents an experimental evaluation centered around software-defined switches and controllers. However, we note the absence of an analytical optimization framework in the work presented. Additionally, the use of Wi-Fi access points as cellular base stations is questionable at best. Tantayakul et al. [97] have presented an OpenFlow-based mobility management framework that is largely focused on reducing packet loss under different mobility conditions. The results presented in [97] primarily ratify the effectiveness of SD-MM schemes.

An on-demand SD-MM scheme has been presented in [98] wherein reduction in signaling costs is achieved through prioritization of delay sensitive flows during the mobility management procedure. However, the presented work only focuses on handovers, ignoring the registration and paging procedures. Yin et al. [99] introduce a hierarchical SD-MM framework that differentiates between handovers that occur within the same control domain, and those that occur across different control domains. In particular, the framework proposed in [99] seeks to optimize inter-domain handovers through the use of a global controller. At the same time we note the global controller concept is unlikely to scale well for large cellular networks.

Furthermore, [100] makes use of distributed hash tables to track user mobility. The proposed framework relies on the presence of a central controller as seen in [99], leading to potential scalability problems. On the other hand, a topology management scheme for real-time notification of changes in the network topology has been proposed in [101]. The work presented in [101] primarily endeavors to track user mobility with the intention of

using the obtained data for efficient resource allocation.

Different from the prior art discussed above, [102] presents an SD-RAN design framework by solving for the optimal number of baseband units (BBUs) needed to serve a given number of remote radio heads (RRHs). The optimization framework presented in [102] is based on the minimization of the monetary costs associated with setting up a BBU and the RRH-BBU latency. While [102] presents a network planning framework with an objective that is similar to this work, we note the following drawbacks in the presented work. First, the system model relies extensively on static parameters, i.e., the link latency is pre-determined, and does not vary with changing link utilization. Second, the problem formulation under consideration does not place a capacity bound on the RRH-BBU links in terms of bandwidth. Third, the impact of the core network has not been taken into consideration. Ee also note that the framework does not involve SD-MM, and therefore mobility management aspects such as registration, paging, and user-specific cluster formation are missing.

Continuing within the broader domain of network design, we note that extensive work has been done in the domain of controller placement for core networks [103, 104, 105, 106, 107, 108, 109], leveraging a variety of optimization techniques ranging from linear programming to multi-objective combinatorial optimization. To this end, we aim to augment these solutions with an equally robust RAN design framework.

In general, we note that the existing solutions are largely focused on the core network, or treat network design and mobility management in isolation, and do not seek to establish a link between the two. However, with the flexibility that SDN and NFV bring to cellular networks, it bodes well to examine the impact of network topology on signaling cost metrics. It is this interdependence that we seek to exploit with the intention of providing an integrated solution.

Table 4.1: Summary of system acronyms used throughout this chapter

Acronym	Description
SDN	Software-Defined Networking
NFV	Network Function Virtualization
SD-RAN	Software-Defined Radio Access Network
DU	Distributed Unit
CU	Central Unit
UE	User Equipment
SD-MM	Software-Defined Mobility Management
OF	OpenFlow
OVS	Open vSwitch
IA-CU	Initial Attach CU
CMR	Call to Mobility Ratio
FADC	Fast-Acting Dynamic Clustering
BNR	Balanced Network Replanning
PPBP	Poisson Pareto Burst Process

4.3 System Model

In this section we provide details concerning the network architecture under consideration, along with the SD-MM framework. Note that in Table 4.1 we provide the notations used throughout this chapter.

4.3.1 Network Architecture

The network model under consideration consists of an SDN controller, OpenFlow (OF) enabled switches, Open vSwitch (OVS) enabled CUs, hardware-only DUs and a certain number of UEs, as shown in Figure 4.1. A change in the serving DU within the domain of the same CU is not considered a location update event, and therefore the associated signaling costs are not taken into consideration. In other words, a UE does not perform registration if there is a change in the serving DU within the same CU.

Futhermore, we assume that the entire network is based on proactive flow instantiation, in which flow tables at each CU are pre-determined. The proactive approach is useful

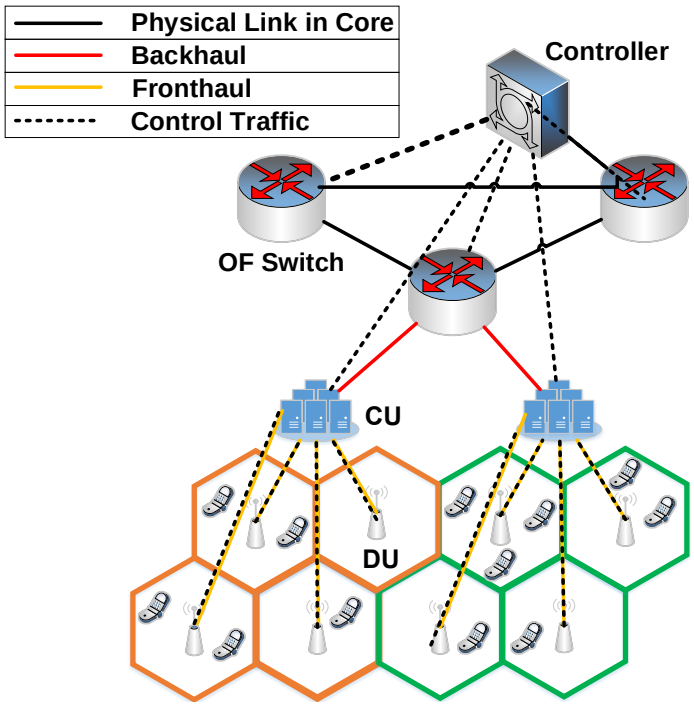


Figure 4.1: Reference network architecture.

because it needs to be performed only once per network planning iteration, i.e., each time the CU configuration in the network is changed. Additional advantages of a proactive approach include reduced latency, while ensuring that there is no disruption even if a controller goes down, so long as the network is not reconfigured.

Additionally, for each UE, i, there exists a virtual cluster of CUs, n_i. For each cluster, the CU at which the UE performs the initial attach procedure serves as an anchor point, and we refer to this CU as the initial attach CU (IA-CU) throughout this work. The cost of signaling over the fronthaul, and between CUs in a cluster has been considered negligible, and therefore does not figure in this work. The entire network modeled as a graph has been shown in Figure 4.2.

4.3.2 Software-Defined Mobility Management Framework

Herein we describe our proposed SD-MM framework and the underlying algorithm. A location update event that is triggered by a change in the serving CU within a cluster results

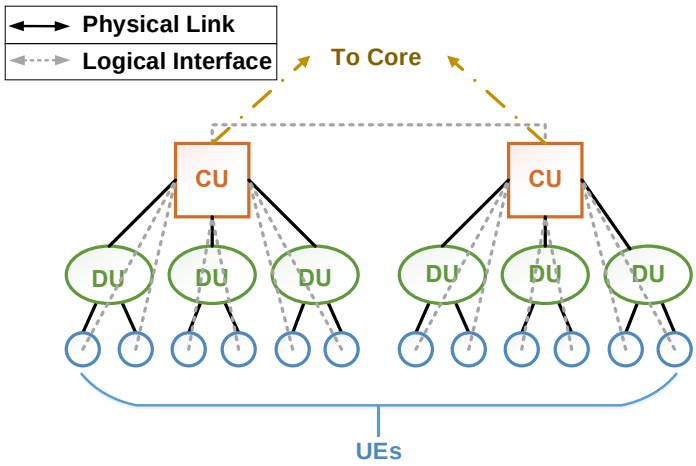


Figure 4.2: Graph representation of network.

in a minimal amount of signaling required to inform the IA-CU of a change in the serving CU. At the outset, we do not consider the location update cost in this scenario. However, in the event that the UE moves to a CU not within its cluster, the controller in the core must be informed of this change which involves a significant cost, as the target CU must now perform signaling over the core network. It is this cost associated with inter-cluster location updates, expressed in terms of the traffic on the links between a particular CU and its corresponding core controller that is of particular interest.

To summarize the location update framework:

- Intra-cluster location updates do not incur a cost.
- A change in the serving CU, such that the target CU lies outside the UE's cluster, incurs a location update cost.

Figure 4.3 provides a graphical representation of this approach. In this figure, there are three different clusters, with each cluster corresponding to a specific user. For example, cluster 1 belongs to UE 1, cluster 2 to UE 2, and so on. As UE 1 moves from one CU to another within cluster 1, it does not incur any location update cost. However, if it moves to a CU outside its cluster, then a location update must be performed.

On the other hand, the paging process involves the controller signaling the IA-CU which

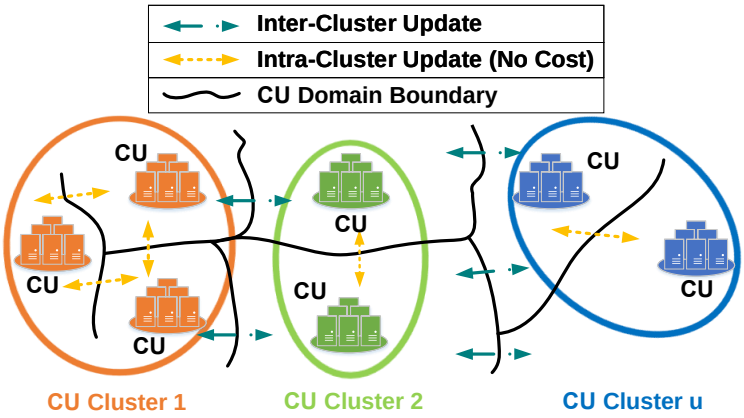


Figure 4.3: Location updates in a clustered arrangement.

then forwards the page to the serving CU. The serving CU then signals a service request to the controller, followed by call establishment. Since, we have considered bandwidth costs of intra-cluster signaling to be minimal, cluster size has an insignificant impact on the bandwidth costs associated with paging, as the only bandwidth consideration is that between the core controller and the IA-CU, which is expected to be a function of the traffic in the core network as outlined in Figure 4.4. In a nutshell, whether we employ clustering or not, the nature of the cost is not impacted. Consequently we are more concerned with the delay associated with cluster size, when it comes to the paging process as further detailed in Section 4.4.

The first mobility metric we define is the call arrival rate. As is standard, we model call arrivals as a Poisson point process, with a mean value of λ_i calls per unit time, for user i [110, §6]. Similarly, we express CU residence time of user i in terms of an exponential distribution, with mean $1/\sigma_i$, therefore, the CU crossing rate, hereby referred to as the mobility rate, takes the form of σ_i CU crossings per unit time. The UE arrival rate at an DU r is expressed by a mean κ_r arrivals per unit time. The serving time associated with the j^{th} CU is modeled as an exponential distribution with mean $1/\mu_j$. Accordingly, its processing capacity is given by μ_j. The CU, by design, is limited by its serving capacity. There exists a logical connection between the j^{th} CU and the c^{th} controller, made possible by a set of $l \in L$ links.

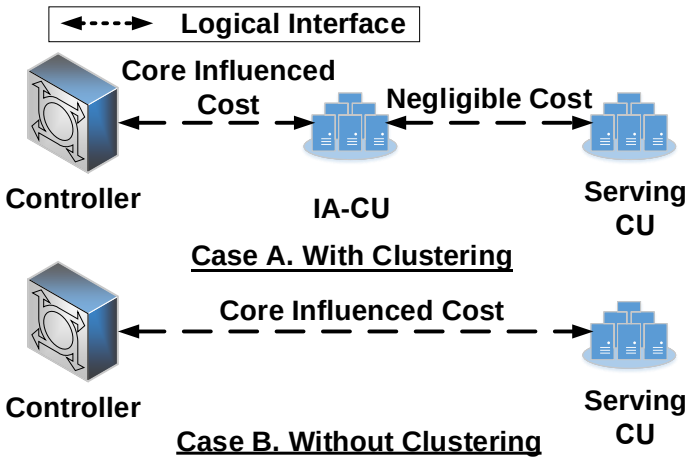


Figure 4.4: Bandwidth cost for paging.

Furthermore, each user $i \in I$ is characterized by a call to mobility ratio (CMR_i) as follows

$$CMR_i = \frac{\lambda_i}{\sigma_i}, \tag{4.1}$$

thus,

$$\lambda_i = CMR_i \sigma_i. \tag{4.2}$$

Since our proposed framework seeks to perform updates in a dynamic manner, two different classes of updates are apparent. The first class of updates, which we will henceforth refer to as fast-acting dynamic clustering (FADC), represents the clustering decision that must be taken each time the user moves out of their cluster or the mobility metrics associated with that user change.

For any user i, FADC is first performed during the initial attach procedure to determine the optimal number of CUs, n_i, in the user's cluster, with the serving CU being set as the IA-CU. Each time the UE changes its serving DU, it checks the physical cell ID (PCI) of the associated CU. For each new PCI encountered, the UE increments a counter $c_{i,PCI}$. When $c_{i,PCI}$ exceeds n_i, it implies the UE has now moved outside its initial cluster, triggering FADC, which recalculates the new optimal cluster size. Additionally, a change in the call arrival rate λ_i or the mobility rate σ_i will trigger FADC. FADC will first calculate the

updated signaling cost for user i, c_i^+, and if the updated cost is found to exceed the previous cost c_i by an amount ϵ, the optimal cluster size will be re-calculated. We note that c_i has been described in further detail in Section 4.4. The ϵ value is set by the system operator depending on the required tolerance. The actions of FADC can be summarized as:

- Calculation of new signaling cost c_i^+, if FADC is triggered by a change in user metrics.
- Calculation of the optimal cluster size, in the event of movement outside cluster or cost deviation beyond threshold.
- Setting the CU where FADC cluster calculation is performed as the IA-CU.

The second class of updates deal with CU deployments and assignments. We refer to these updates as balanced network replanning (BNR). The rationale behind the introduction of BNR is that, it is not feasible to perform a network-wide planning update each time the metrics associated with a single user change, as doing so would introduce significant overhead. Instead, for each FADC event that is triggered, BNR calculates the updated overall system cost, C_T^+. In the event that the system cost exceeds the existing cost C_T by a margin ε, a network-wide planning update is triggered, including cluster calculation for all UEs. The actions performed by BNR can be summarized as:

- Calculation of the optimal number of CUs.
- Determination of DU-CU and CU-Controller assignments.
- Calculation of the optimal cluster size, and IA-CU initialization for all users.

In this manner, FADC can be viewed as a subset of the actions performed by BNR. A structured approach of this kind helps prevent unnecessary overhead each time a user re-adjusts their cluster. Additionally, we expect that system operators would prefer to use a suitably large value for ε to prevent BNR from being triggered too frequently. The associated process flow has been outlined as part of Algorithm 1. Furthermore, we note that SDN and

Algorithm 1 Proposed BNR-FADC Framework

for $i \in I$ **do**
  if change in serving DU **then**
    check PCI of CU associated with new DU
    if new PCI encountered **then**
      $c_{i,PCI} \leftarrow c_{i,PCI} + 1$
      if $c_{i,PCI} > n_i$ **then**
        perform FADC
        calculate C_T^+
        if $C_T^+ > C_T + \varepsilon$ **then**
          perform BNR
        end if
      else
        update serving CU at IA-CU
      end if
    end if
  else if change in λ_i or σ_i **then**
    calculate c_i^+
    if $c_i^+ > c_i + \epsilon$ **then**
      perform FADC
      calculate C_T^+
      if $C_T^+ > C_T + \varepsilon$ **then**
        perform BNR
      end if
    end if
  end if
end for

NFV are vital to the success of the proposed BNR-FADC framework as they allow for seamless CU migration, along with on-demand changes in cluster configuration.

Next, we need to consider the trade-off between registration and paging. An optimal location update strategy will seek to group all CUs together in one large cluster, so that the location update message is not transmitted to the core. This assertion falls in line with the system model described above. On the other hand, an optimal paging strategy will always favor reporting a CU change to the core controller, such that paging cost is minimized. Therefore, a joint optimization framework is needed.

4.4 Problem Formulation

As outlined in Section 4.1, we seek to determine the optimal number of CUs required and the corresponding DU-CU and CU-controller associations for a given set of UE mobility metrics, existing DUs, and core network availability. Furthermore, we also endeavor to define registration and paging costs and determine the number of CUs in user-specific clusters with the intention of minimizing the aforementioned costs. The RAN we have considered has a total of R DUs available, and each DU $r \in R$ has equal processing capacity. Typically a CU j would be co-located with a DU site, i.e., $j \in J \subseteq R$ such that

$$y_r = \begin{cases} 1, \text{ if CU } j \text{ is co-located with DU } r; \\ 0, \text{ else,} \end{cases} \tag{4.3}$$

and the total number of CUs is given as

$$B = \sum_{r \in R} y_r, \tag{4.4}$$

with B_{MAX} representing the maximum acceptable number of CUs in the system. To ensure that a DU is assigned to only one CU at a time, we define

$$x_{rj} = \begin{cases} 1, \text{ if DU } r \text{ is assigned to CU } j; \\ 0, \text{ else.} \end{cases} \tag{4.5}$$

Furthermore, to maintain a certain level of quality-of-service (QoS) for UEs connecting to each DU, we define a target latency δ_{rj} for each DU-CU pair, such that

$$\delta_{rj} = \begin{cases} 1, \text{ if } latency(r,\ j) \leq delay_{TH1}; \\ 0, \text{ else,} \end{cases} \tag{4.6}$$

where $delay_{TH1}$ is an operator set threshold. Combining (4.5) and (4.6), we have

$$\sum_{j \in J \subseteq R} x_{rj}\delta_{rj} = 1 \qquad \forall\, r \in R. \tag{4.7}$$

Next, we also need to ensure that a DU r is assigned to a CU j, if and only if CU j exists, i.e.,

$$y_r - x_{rj} \geq 0 \qquad \forall\, r \in R,\ j \in J \subseteq R. \tag{4.8}$$

The constraint above eliminates the possibility of rogue assignments because it will be satisfied only if a CU j exists, and a DU r is assigned to it

With the DU-CU assignments defined, we now perform assignments between the CUs in the RAN, and the controllers in the core via the OVSs. In general, if a CU j is assigned to a controller c we have

$$x_{jc} = \begin{cases} 1, \text{ if CU } j \text{ is assigned to controller } c; \\ 0, \text{ else.} \end{cases} \tag{4.9}$$

Ensuring that only a single controller c is associated with a CU j, and the assignment satisfies a delay bound, it follows that

$$\sum_{c \in C} x_{jc}\delta_{jc} = 1 \qquad \forall\, j \in J \subseteq R, \tag{4.10}$$

where

$$\delta_{jc} = \begin{cases} 1, \text{ if } latency(j, c) \leq delay_{TH2}; \\ 0, \text{ else,} \end{cases} \tag{4.11}$$

and $|\mathrm{C}|$ is the optimal number of controllers obtained from the Optimal Multi-Controller Placement scheme outlined in [109]. Furthermore, the $delay_{TH2}$ parameter must be specified by the system operator based on their QoS requirements. Finally, this assignment is

constrained by the capacity, μ_c, of each controller c as follows

$$\sum_{j \in J \subseteq R} x_{jc} u_j < u_c \qquad \forall\, c \in C. \tag{4.12}$$

At this stage, we have assigned DUs to CUs, and CUs to the controllers of a core network that has already been deployed. On the other hand, we must also determine the number of CUs that will form a part of each per-user cluster, so as to minimize the overall signaling costs. It may be noted that cluster formation, like DU assignment, is a dynamic process.

As noted earlier, the CU j has a limited processing capacity given by μ_j. In the same vein, let us consider the different signaling flows that must be handled by a single CU:

- Location Update events of each UE i, arriving at DU r of CU j such that $j \notin N_i$.
- Paging events of each UE i served by CU j, in addition to the UEs for which CU j is the IA-CU.

The association of a CU j with the cluster of UE i is expressed as

$$t_{ij} = \begin{cases} 1, \text{ if CU } j \text{ is present in the cluster of UE } i; \\ 0, \text{ else,} \end{cases} \tag{4.13}$$

such that

$$\sum_{j \in J} t_{ij} = n_i \qquad \forall\, i \in I. \tag{4.14}$$

Now, we must define the two signaling flows identified in the previous paragraph. At a CU j, the registration flow, r_j, originating from newly arrived UEs, for which said CU assumes the role of an IA-CU is

$$r_j = \sum_{r \in R} \kappa_r \left(\frac{\sum_{i \in I} (1 - t_{ij})}{B_{MAX}} \right) x_{rj} \delta_{rj} \quad \forall\, j \in J \subseteq R. \tag{4.15}$$

It may be noted that the above equation is representative of the flow from a CU-to-

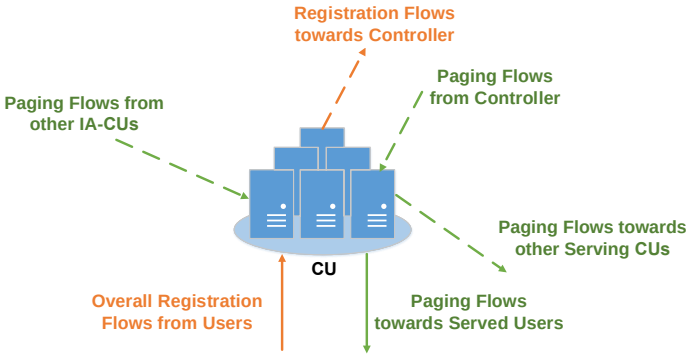


Figure 4.5: Signaling flows at the CU.

controller perspective as opposed to the user's perspective. Next, we assume that the paging flow originates at the controllers in the core, and the flow arriving at CU j from controller c has two components of the form:

- IA-Component for those UEs for which the CU is an IA-CU, i.e., this flow will be forwarded to the respective serving CUs.
- Native component for those UEs for which the CU is the serving cell.

The presence of the IA component further leads to another interesting observation. In addition to the two controller originating flows mentioned above, there exists a third type of control flow in the network. As each CU forwards the IA component to the respective serving CU, at any given CU we witness the arrival of several flows originating from other CUs. These paging flows are for those UEs for which the given CU is not the IA-CU, but is the serving CU for the current network state. The incoming and outgoing flows at a CU can be thus modeled as shown in Figure 4.5.

A clustering environment necessitates the presence of the IA-component because a CU not only processes flows for the UEs it serves but also forwards flows for those UEs for which it is the IA-CU. Thus, a CU processes the paging flows for: (i) UEs for which it is both the IA and serving CU, (ii) UEs for which it is the IA-CU only, and (iii) UEs for which

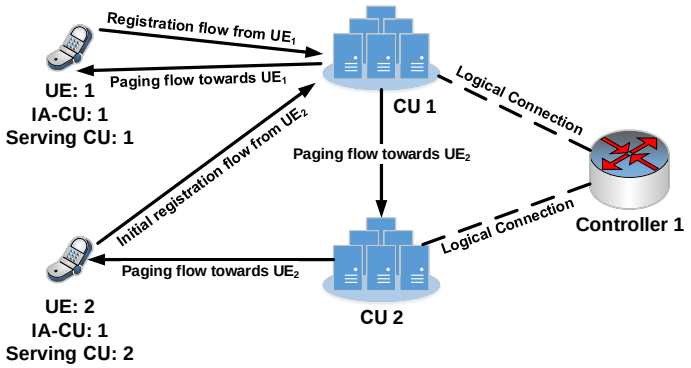


Figure 4.6: Example scenario depicting registration and paging flows.

it is the serving CU only. Accordingly, the paging flow at CU j, p_j, can be expressed as

$$p_j = 2 \sum_{i \in I} CMR_i \sigma_i t_{ij} \quad \forall\, j \in J \subseteq R. \tag{4.16}$$

Figure 4.6 depicts an example scenario showing the possible registration and paging flows at the CU. The serving capability of each CU must be sufficient to at least handle both the signaling flows described above

$$r_j + p_j < u_j \qquad \forall\, j \in J \subseteq R. \tag{4.17}$$

With clustering in place, we seek to develop formulations for costs associated with the registration and paging procedures. Since the cost of signaling over the fronthaul and between the CUs has been considered negligible, the bandwidth cost of a single location update can be represented as function of the traffic on the path traversed by said update, between any given CU and its corresponding core controller. From the perspective of the core network, the registration flow originates at the CU and paging flow originates at the controller. Both kinds of control flows traverse the same link, but the direction of traversal is opposite. Therefore, the next step is to determine the control traffic flowing over the links. In general, we assume that while the two signaling flows are represented independently, the

links in the core network are symmetric and bidirectional, i.e., an upstream path from the CU to its controller implies the existence of a corresponding downstream path.

We define three traffic assignment matrices. Two of these, the registration traffic matrix $\boldsymbol{Z_{REG}} = [z_{jcl}^{REG}]\ \forall\ j \in J \subseteq R, c \in C, l \in L$, and the paging traffic matrix $\boldsymbol{Z_{PAGE}} = [z_{jcl}^{PAGE}] \quad \forall\ j \in J \subseteq R, c \in C, l \in L$, where $[z_{jcl}^{REG}]$ represents the registration traffic between CU j and controller c, over link l, and $[z_{jcl}^{PAGE}]$ represents the paging traffic between controller c and CU j over the same link. The third traffic assignment matrix is based on the forwarded traffic that a CU receives from other CUs, and is represented by $\boldsymbol{Z_{FWD}} = [z_{j'jl}^{FWD}] \quad \forall\ j', j \in J \subseteq R, l \in L$. Furthermore, we consider there exist Q_{jc} paths between a CU j and its corresponding controller c, and, we define a topology control matrix $\boldsymbol{C_{jc}}$ having $|L|$ rows and $|Q_{jc}|$ columns as

$$\boldsymbol{C_{jc}}[l, q] = \begin{cases} 1, \text{ if the } l\text{th link serves the } q\text{th path;} \\ 0, \text{ else.} \end{cases} \tag{4.18}$$

The paths should be free from loops, and unique, i.e., it should not be possible to express any one path as a linear combination of two or more other paths, therefore the matrix $\boldsymbol{C_{jc}}$ should always be full column rank [111].

Next, we must obtain the different paths a link l supports in terms of control flows. Taking the left inverse of $\boldsymbol{C_{jc}}$ and multiplying it by the lth standard basis, e_l, gives us a column vector c_{jcl}, i.e., $c_{jcl} = \boldsymbol{C_{jc}}^{-1} e_l$ where $\boldsymbol{C_{jc}}^{-1} = (\boldsymbol{C_{jc}}^T \boldsymbol{C_{jc}})^{-1} \boldsymbol{C_{jc}}^T$. It may be noted that the left inverse of $\boldsymbol{C_{jc}}$ exists because it is full column rank. Taking the 1-norm of c_{jcl}, we obtain

$$g_{jcl} = \|c_{jcl}\|_1, \tag{4.19}$$

where g_{jcl} represents the sum of the number of paths between CU j and controller c that use link l. By performing the same set of operations on $\boldsymbol{C_{j'j}}$, we can obtain the number of

paths between the same CU j and some other CU j', that make use of said link, as

$$g_{j'jl} = \|c_{j'jl}\|_1. \tag{4.20}$$

Invoking the flow conservation constraint at CU j

$$\sum_{l \in L} g_{jcl}(z_{jcl}^{REG} + z_{jcl}^{PAGE}) \; + \sum_{\substack{j' \in J \\ j' \neq j}} \left(\sum_{l \in L} g_{j'jl} z_{j'jl}^{FWD} \right) = r_j + p_j$$

$$\forall \; j, j' \in J \subseteq R, c \in C. \tag{4.21}$$

Similarly, the overall average signaling traffic carried by link l is expressed as

$$s_l = \sum_{j \in J \subseteq R} \sum_{c \in C} (z_{jcl}^{REG} + z_{jcl}^{PAGE}) \; + \sum_{\substack{j \in J \subseteq R \\ j \neq j'}} \left(\sum_{j' \in J \subseteq R} z_{j'jl}^{FWD} \right). \tag{4.22}$$

With the control traffic assignment complete, we now turn our attention to data traffic. Data traffic is bursty in nature, and exhibits long range dependence [112], which is characterized by a power-like decay of the auto-correlation function. Accordingly, we model data traffic as a self-similar process, having a Hurst exponent between 0.5 and 1. For this purpose, we choose the Poisson Pareto burst process (PPBP) as it satisfies both long range dependence and self-similarity [113].

The PPBP is a process based on the overlapping of multiple bursts within a discrete time interval, with the burst durations being characterized by a long-tailed distribution, the Pareto distribution in particular. A rather detailed treatment of the PPBP and its applicability to data traffic modeling can be found in [114]. Drawing on the analysis carried out in [114], a discrete time traffic model is used, i.e., the overall time-scale can be divided into a finite number of sampling intervals. Within each sampling interval, the arrival of data traffic is assumed to follow a continuous time model. For each link l, burst arrivals are modeled according to a Poisson process with rate λ_l. The Pareto distributed burst lengths

are characterized by the Hurst exponent H and a mean value ϖ. Furthermore, each burst is represented by a static flow rate, η. It may be noted that we assume all bursts to have the same characteristics in terms of H, ϖ and η.

The probability density function of the burst length ς is expressed as

$$p_{\varsigma}(x) = \frac{\alpha k^{\alpha}}{x^{\alpha+1}}, \qquad x \geq k > 0, \tag{4.23}$$

and the corresponding cumulative distribution function as

$$F_{\zeta}(x) = 1 - \left(\frac{k}{x}\right)^{\alpha}, \qquad x \geq k > 0, \tag{4.24}$$

where k specifies the start of the Pareto tail, and α is the shape factor specifying its rate of decay. If $\alpha \leq 2$, the distribution has infinite variance, and if $\alpha \leq 1$, it has infinite mean [110], since working with infinite duration bursts is not tractable, we set $1 < \alpha < 2$, thus

$$\varpi = E[\zeta] = \frac{k\alpha}{\alpha - 1}. \tag{4.25}$$

Given the mean burst duration, the average number of active bursts on a link l can be derived as $n_l = \lambda_l \varpi \; \forall \; l \in L$, and for a flow rate η, the data traffic on the same link can be expressed as $\tau_l = N_l \eta = \lambda_l \varpi \eta \; \forall \; l \in L$.

At this stage, it may be noted that both control and data traffic are multiplexed onto the same link. Consequently, the average traffic on link l can be expressed as $z_l = s_l + \tau_l$, with the link capacity constraint being given by $z_l < \mu_l$. Furthermore, we define the link traffic matrix $\boldsymbol{Z_l} = [z_l] \; \forall \; l \in L$, with size $|L|$ x $|1|$.

Now, we derive the cost of a single location update, c_{UPD}. For this purpose, the matrix $\boldsymbol{C_{jc}}$ described earlier will come into play. Each column of $\boldsymbol{C_{jc}}$ represents a possible path Q_{jc}, and the links used by that path. The signaling cost over each path can obtained by taking the transpose of the product of $\boldsymbol{C_{jc}}$ and the qth standard basis, where q is the path index, i.e.,

$q = 1, \ldots, |Q_{jc}|$, and multiplying the result by $\boldsymbol{Z_l}$. In other words, the cost incurred over a single path is given by $(\boldsymbol{C_{jc}} e_q)^T \boldsymbol{Z_l}$, extending the same idea to $|Q_{jc}|$ paths, we obtain the cost of a single location update

$$c_{UPD} = \sum_{q=1}^{|Q_{jc}|} (\boldsymbol{C_{jc}} e_q)^T \boldsymbol{Z_l}. \tag{4.26}$$

Next, we must quantify the registration instances triggered by user i. Operating in a clustered arrangement, a user must move through n_i unique CUs in order to trigger a location update. Given a mobility rate of σ_i, we calculate the probability of moving across at least n_i clusters, and use that to express the effective cost of registration per user, $c_{Li} \; \forall \; i \in I$, as

$$c_{Li} = \begin{cases} \left(\left(\sum_{n=n_i}^{\sigma_i} \binom{\sigma_i}{n} (1-p_i)^n (p_i)^{\sigma_i - n} \right) \sigma_i \right) c_{UPD} \\ \qquad\qquad\qquad\qquad \text{if } \sigma_i > n_i - 1; \\ 0, \quad \text{otherwise,} \end{cases} \tag{4.27}$$

where $p_i = \frac{n_i}{B_{MAX}}$, with the overall location update cost across all I UEs being expressed as $C_L = \sum_{i \in I} c_{Li}$. At the outset (4.27) outlines the advantage of clustering. By grouping together n_i CUs, we are able to effectively reduce the mobility rate as all the CUs within a cluster now identify as a single CU. Thus, larger is the cluster size, lower will be the individual location update cost. As outlined earlier, the paging cost is modeled in terms of delays, from the instance the controller issues the page, to when the serving CU receives the paging command. As a first step, we need to consider the average delay over a single link, which in general can be expressed as

$$\text{Delay over Link} = \frac{\text{Mean Packet Size}}{\text{Available Link Bandwidth}}, \tag{4.28}$$

since we expect all paging flows to be characterized by the same mean packet size, we

Table 4.2: Summary of problem parameters and decision variables for the SD-MM RAN design problem

Symbol	Description
$r \in R$	Index for DUs in the system
$j \in J \subseteq R$	Index for CUs in the system
$i \in I$	Index for UEs in the system
$c \in C$	Index for controllers in the system
$l \in L$	Index for links in the system
$B_{MAX} \geq 0$	Maximum number of allowed CUs
$delay_{TH1} \geq 0$	Latency threshold between DU and CU
$delay_{TH2} \geq 0$	Latency threshold between CU and controller
$\sigma_i \geq 0$	CU crossing per unit time for user i
$\lambda_i \geq 0$	Mean call arrival rate for user i
$\kappa_r \geq 0$	Mean user arrival rate at DU r
$n_i \geq 0$	Number of CUs in the cluster of user i
$u_j \geq 0$	Processing capacity for CU j
$u_c \geq 0$	Processing capacity for controller c
$u_l \geq 0$	Link capacity for link j
$z_l \geq 0$	Average traffic on link l
$c_{Li} \geq 0$	Location update cost for user i
$c_{Pi} \geq 0$	Paging cost for user i
$z_{jcl}^{REG} \geq 0$	Registration traffic between CU j and controller c over link l
$z_{jcl}^{PAGE} \geq 0$	Paging traffic between CU j and controller c over link l

Symbol	Description
$z_{j'jl}^{FWD} \geq 0$	Forwarded traffic between CUs j and j' over link l
g_{jcl}	Sum of number of paths between CU j and controller c that use link l
$g_{j'jl}$	Sum of number of paths between CU j and CU j' that use link l
y_r	Binary decision variable that takes value 1 if CU is deployed at DU r, 0 otherwise
x_{rj}	Binary decision variable that takes value 1 if DU r is assigned to CU j, 0 otherwise
x_{jc}	Binary decision variable that takes value 1 if CU j is assigned to controller c, 0 otherwise
t_{ij}	Binary decision variable that takes value 1 if CU j is present in cluster of user i
δ_{rj}	Indicator variable that takes value 1 if $latency(r,j) \leq delay_{TH1}$, 0 otherwise
δ_{jc}	Indicator variable that takes value 1 if $latency(j,c) \leq delay_{TH2}$, 0 otherwise

perform a slight modification to the above expression, characterizing the average delay over link l, D_l, as

$$d_l = \frac{1}{u_l - z_l} \qquad \forall\, l \in L. \tag{4.29}$$

To make the analysis over the next few steps tractable, we define a link delay matrix $\boldsymbol{D_l}$ of the form $diag(d_1, d_2, d_3, \ldots, d_L)$. The delay over any given path q from a CU to a controller, or q' from an IA-CU to the serving CU can be represented as $\left\|(\boldsymbol{C_{jc}}e_q)^T \boldsymbol{D_l}\right\|_1$ and $\left\|(\boldsymbol{C_{j'j}}e_q)^T \boldsymbol{D_l}\right\|_1$ respectively. Furthermore, since we are employing multi-path routing, the path with the largest delay serves as a bottleneck and determines the location update cost

for a single paging instance in a clustering environment as

$$c_{PAGE} = \max_{q \in Q_{jc}} \{f_q \left\| (\boldsymbol{C_{jc}} e_q)^T \boldsymbol{D_l} \right\|_1\} \quad + \max_{q' \in Q_{j'j}} \{f_{q'} \left\| (\boldsymbol{C_{j'j}} e_q)^T \boldsymbol{D_l} \right\|_1\}, \tag{4.30}$$

where $\sum_{q \in Q_{jc}} f_q = 1$ and $\sum_{q' \in Q_{j'j}} f_{q'} = 1$. Furthermore, if for any UE, the IA-CU is the serving CU, then the matrix $\boldsymbol{C_{j'j}}$ will not exist, and thus the overall paging cost will not be impacted by the second half of (4.30). Therefore, through (4.30) we express the additional delay introduced as a result of clustering.

Going back to the user metrics defined earlier, and making use of the newly defined paging cost, we define the paging cost for a single user as $c_{Pi} = \lambda_i c_{PAGE} \; \forall \; i \in I$, with the total paging cost for the entire network being expressed as $C_P = \sum_{i \in I} c_{Pi}$, and the per-user signaling cost being given by $c_i = c_{Li} + c_{Pi} \; \forall \; i \in I$, with overall signaling cost $C_T = C_L + C_P$.

The optimization framework can thus be summarized as follows, with the problem parameters and decision variables summarized in Table 4.2.

Find: $y_r,\ x_{rj},\ x_{jc},\ n_i,\ t_{ij}$

Minimize: $w_1 \dfrac{B}{B_{MAX}} + w_2 \dfrac{C_L}{C_{LMAX}} + w_3 \dfrac{C_P}{C_{PMAX}}$

Subject To: $\sum_{j \in J \subseteq R} x_{rj} \delta_{rj} = 1 \quad \forall\, r \in R,\ y_r - x_{rj} \geq 0 \quad \forall\, r \in R, \forall\, j \in J,\ \sum_{c \in C} x_{jc} \delta_{jc} = 1 \quad \forall\, j \in J,$

$\sum_{j \in J} t_{ij} = n_i \quad \forall\, i \in I,\ n_i \leq \sigma_i \quad \forall\, i \in I,\ \sum_{j \in J} x_{jc} u_j < u_c \quad \forall\, c \in C,\ r_j + p_j < u_j \quad \forall\, j \in J,$

$z_l < u_l \quad \forall\, l \in L,\ \sum_{l \in L} g_{jcl} (z_{jcl}^{REG} + z_{jcl}^{PAGE}) + \sum_{\substack{j' \in J \\ j' \neq j}} \left(\sum_{l \in L} g_{j'jl} z_{j'jl}^{FWD} \right) = r_j + p_j \quad \forall\, j \in J,$

where C_{LMAX} and C_{PMAX} are appreciably high registration and paging costs used as normalizing parameters for the objective, with $0 \leq w_1,\ w_2,\ w_3 \leq 1$ and $w_1 + w_2 + w_3 = 1$.

The problem presented above is characterized by a large number of variables and a complicated structure that makes it intractable. Thus, we perform a set of pre-processing steps to simplify the problem. We note that since CU positions in the network are a subset

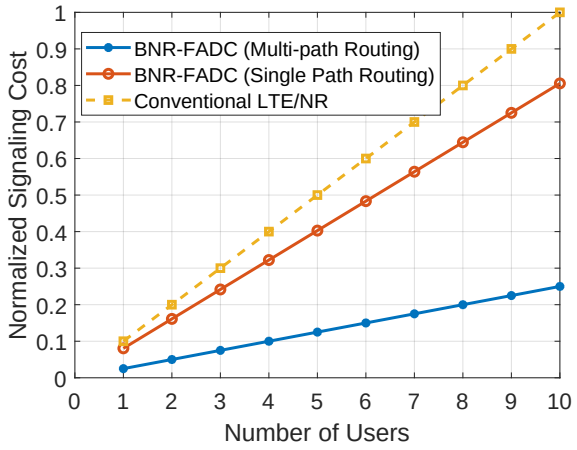


Figure 4.7: Variation in signaling cost with change in the number of users.

of the DU positions, for each DU position r, we determine a candidate controller c which is characterized by the shortest path from r to c. Furthermore, since multi-path routing is in use, we also determine $k_r - 1$ additional shortest paths from r to c, where the cost associated with a path k is given by $\sum_{l \in k} \tau_l$, along with $k_{rr'}$ shortest paths from DU r to every other DU r'. Thus, if a CU j is deployed at DU r, it is constrained to use the candidate controller c, and the k_r and $k_{rr'}$ paths determined previously. Furthermore, the signaling traffic distribution over these paths is in inverse proportion to the path cost $\sum_{l \in k} \tau_l$.

In doing so, we note the following:

- First, we no longer need to explicitly solve for x_{jc}, since the controller assignment now solely depends on the DU at which a CU is deployed.

- Second, g_{jcl}, z_{jcl}^{REG}, z_{jcl}^{PAGE}, $g_{j'jl}$, and $z_{j'jl}^{FWD}$ can now be determined in accordance with the pre-processing procedure.

With the problem formulation and pre-processing in place, we proceed to evaluate the performance of our system as is outlined in the next section.

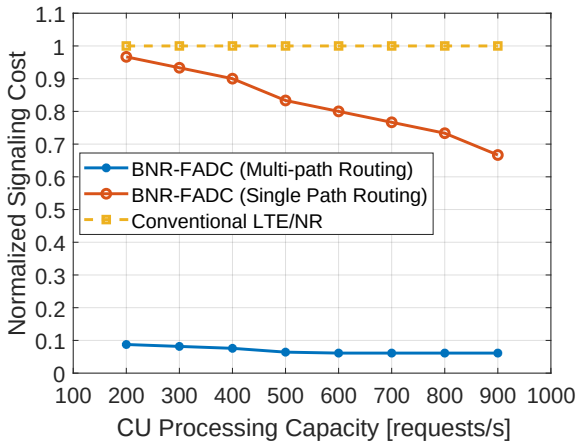


Figure 4.8: Signaling cost comparison for changing CU processing capacity.

4.5 Performance Evaluation

As part of the system performance evaluation, we focus on the single most important metric– the signaling cost, which in this case is given by the sum of normalized registration and paging costs. Using the MATLAB [115] platform for performance evaluation, the network topology under consideration consists of 2 controllers, a total of 5 DUs, and a maximum of 10 UEs. Furthermore, B_{MAX} is set equal to the number of DUs in the system, and all entity groups in the system, i.e., UEs, DUs, CUs, links, and controllers share the same parameters across the group. We begin by examining the impact of an increasing number of UEs in the system, followed by the system response to changes in CU capacity, link capacity, and link delay levels.

First, we note that Figure 4.7 represents the change in signaling costs as the number of UEs in the system increases from 1 to 10. Furthermore, we consider three cases for evaluation– conventional LTE/NR, and BNR-FADC with and without multi-path routing. Within this context, conventional LTE/NR does not employ clustering, and uses single path routing only. At the outset, BNR-FADC with multi-path routing offers the best performance, outperforming the conventional non-clustering system by nearly 80%. In particular, as the results show, the advantages of clustering become apparent as the number of UEs increases. Additionally, we see that the multi-path routing system outperforms the single path case by

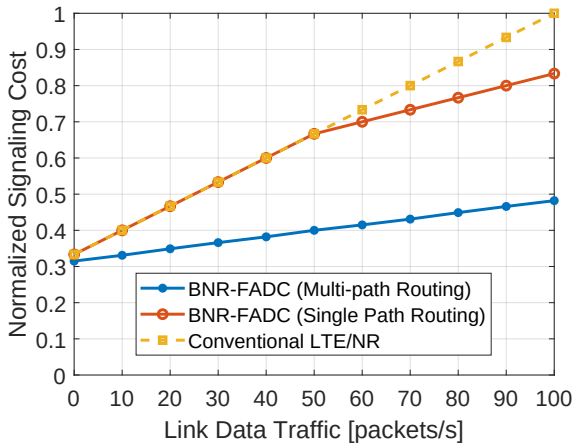


Figure 4.9: Variation in signaling cost with change in link data traffic.

nearly 65% in the 10 UE test scenario. A result of this kind is expected due to the fact that the use of multi-path routing allows the system to distribute signaling traffic across multiple links resulting in load balancing that helps avoid too high of a link utilization, z_l, for any single link.

Next, we argue that a significant constraint in the clustering process is the available processing capacity or the level of congestion at the CUs. A highly congested CU with low available processing capability will not allow the system to exploit the full advantage of the cost reduction offered by clustering, on the other hand, if the CU capacity is not a constraint in deployment, the potential cost reduction can be significant. However, in a real world deployment scenario, CU congestion is a very practical concern. A decrease in the per-CU clustering capacity would have two effects: (i) it would necessitate the deployment of a larger number of CUs and (ii) result in a reduction in the per-user cluster size. At the same time, the conventional system will respond to a decrease in processing capacity by simply increasing the number of CUs.

Figure 4.8 quantifies the impact of CU processing capacity on signaling cost. The results have been obtained by varying the average CU processing capacity, u_j, from 200 requests/s to 900 requests/s in steps of 100. At the outset, we note that the conventional system displays a fixed maximum cost throughout, while the two BNR-FADC schemes demonstrate

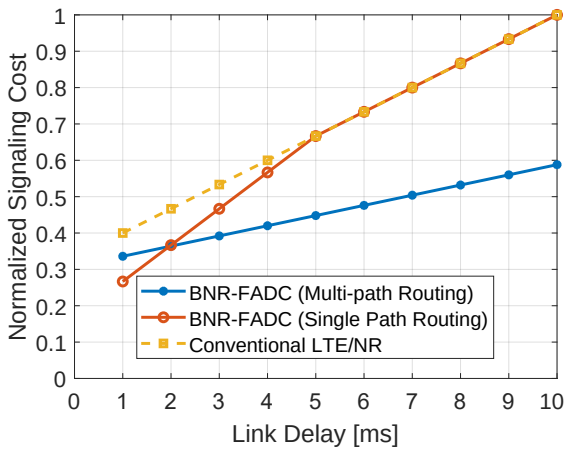


Figure 4.10: Variation in signaling cost with change in link delays.

decreasing costs with increasing capacity. Of particular interest here is the BNR-FADC system with multi-path routing. We note that the signaling cost in this case successively decreases from 0.1 at 200 requests/s to 0.06 at 600 requests/s before plateauing. Upon further examination, we note that this result can be attributed to the fact that the number of CUs decreases from 5 to 1 correspondingly. A larger number of CUs with lower capacity results in smaller per-user clusters, thus increasing the signaling instances. On the other hand, fewer high-capacity CUs can serve as cluster constituents for a larger number of users, resulting in a decrease in the signaling costs.

Additionally, we also examine the impact of changing link metrics on the system. First, we take a look at the variation in signaling costs with change in data traffic, τ_l, on the link from 0 packets/s to 100 packets/s as shown in Figure 4.9. Here too, the advantage of BNR-FADC with multi-path routing is apparent, offering nearly 50% better performance in the highly congested case. Interestingly, the single path routing case follows the results of the conventional solution till $\tau_L = 50$ packets/s, with this result being attributed to the fact that $n_i = 1 \; \forall \; i \in I$ up to this point, and therefore the single-path system mirrors the conventional system's performance. A similar trend is also observed in Figure 4.10, where the link delay, $1/u_l$, is varied from 1 ms to 10 ms. In this case, single-path routing performs better for low-latency links, however as the link capacity falls and delays increase,

this performance advantage falls rapidly with the multi-path system once again offering the best performance.

4.6 Highlights

In this chapter we have addressed the CU planning and clustering problem augmented with software-defined mobility management. We have presented detailed analytical characterizations of user mobility, system traffic, and signaling costs, using them to obtain the least cost solutions for a variety of scenarios. Furthermore, the performance evaluation presented confirms the efficacy of our solution compared to traditional LTE/NR networks. To this end, we hope that this work will serve as a benchmark in the design of software-defined radio access networks.

CHAPTER 5
ADAPTIVE CONTAINERIZATION FOR MICROSERVICES-BASED CORE NETWORKS

Thus far, we have presented a variety of novel solutions that are geared towards the access network, ranging from AirHYPE in Chapter 3 to the SD-MM based design framework in Chapter 4. However, in addition to the RAN, the core network forms a vital component of the cellular infrastructure too. With a view to complementing the Service-Based Architecture (SBA) introduced by 3GPP for 5G and beyond core networks, this chapter presents a novel containerization framework for microservices within the core.

5.1 Motivation and Related Work

In recent years there has been a paradigm shift in software architecture design from an on-premises model to a cloud-native approach [116]. This change has been largely driven by the demand for greater system reliability, scalability, and flexibility. However, traditional applications are monolithic in nature, i.e., built as a single unit, primarily consisting of a database, a client-side interface, and a server-side application. This monolithic architecture suffers from a number of drawbacks:

- System updates are cumbersome as the developer must re-deploy the entire server-side application each time.
- Failure diagnosis is challenging since the monolithic application operates as a single unit.
- Scalability is difficult to achieve leading to resource wastage.

Consequently, a monolithic application architecture cannot leverage the benefits of a cloud-native approach. To this end, the microservices architecture [117] has emerged as

a suitable candidate for cloud-based software deployment. The microservices approach is based on the decomposition of complex software systems into multiple independent services, each with its own system logic and data store. With a view to leveraging the benefits associated with this paradigm shift, the cellular core has witnessed a recent shift from an application-based architecture to a cloud-native microservices-based design [118], with different core network applications being implemented as microservices.

Microservices can be deployed independent of each other and communicate either through standardized interfaces or event messages, making them amenable to deployment through virtualization. Resource virtualization can be achieved primarily through the use of either hypervisors or containers [119]. Hypervisor-based virtualization allows for the provisioning of multiple isolated virtual machines (VMs) over the same physical hardware [120, §2.3.3]. This approach allows each VM instance to have its own unique operating system (OS) with its own binaries, libraries, and applications. Hypervisors can further be categorized into: (i) Type 1 hypervisors that operate on top of the host's hardware, and (ii) Type 2 hypervisors that operate on top of the host's OS. The principal drawback of the hypervisor-based approach is the high overhead associated with maintaining a full OS within each VM instance.

On the other hand, containerization [121] is a form of virtualization that attempts to achieve resource isolation with minimal overhead by sharing the kernel with the host OS. The use of containerization allows us to implement many of the desirable features associated with cloud platforms– elasticity, reliability, and ease of management. Therefore, we focus on container-based virtualization in this chapter. Microservices can be conveniently packaged into containers that are then deployed onto physical hardware, thus ensuring a consistent software execution environment from the developer to the consumer. The use of containerization also allows for inherent scalability and creates a redundancy mechanism for machine failure as container instances can be added and removed on demand.

While the decomposition of a monolithic core network application into a set of microser-

vices is the prerogative of the application developer, the deployment of microservices onto physical hardware is a run-time exercise. Traditionally, microservices have been deployed on single-vendor clouds in accordance with their resource requirements without much regard for either resource or cost optimization. However, the emergence of Cloud Brokers [122, 123] has provided consumers with the opportunity to optimize their usage of cloud infrastructure. More specifically, within the context of the core network, the consumers are the communications service providers (CSPs). The Cloud Broker serves as an intermediary between the Cloud Consumers or CSPs who wish to deploy network applications, and the Cloud Providers who own the underlying infrastructure, as described in Section 5.2.

Accordingly, there have been a number of attempts at performing optimal deployment of microservices over cloud infrastructure [124, 125, 126, 127, 128, 129]. More specifically, the work presented in [124] proposes a dynamic CPU resource allocation framework for Docker containers that aims to reduce resource over-utilization. However, while the presented framework implements adaptive control, it is restricted to the management of computing resources only, and does not consider other vital metrics associated with service-level agreement (SLA) requirements such as those related to latency or reliability. In a similar vein, the framework introduced in [125] presents a linear programming based approach to microservice deployment using Docker. However, here too, the presented model does not take into account service reliability. Within the context of this chapter, we define reliability as a measure of the hardware infrastructure failure rate.

Furthermore, the authors in [126] have presented a genetic algorithm for optimizing container allocation in the cloud, however, the model presented in [126] does not take into account latency considerations associated with service delivery. Latency constraints are essential for ensuring timely delivery of services. In addition, we note that the large convergence time of genetic algorithms is undesirable in the context of run-time resource allocation. At the same time, [127] can be regarded as one of the more comprehensive works in this domain, presenting both a container as well as a VM model, along with limited

support for quality of service (QoS) metrics.

On the other hand, the SmartVM platform introduced in [128] puts forth a two-tier classification of microservices based on their functional logic and performs dynamic container resource management in a manner that satisfies the applications' QoS requirements. However, we note that [128] does not discuss the mapping of containers to the underlying physical substrate. In [129], S. Chen et al. introduce QoS-aware resource partitioning for microservices, allowing latency-critical microservices to share the same physical node while still meeting their QoS requirements. While the presented work is pioneering in terms of resource allocation within a single physical machine, it does not delve into resource allocation across a cluster of such machines. We note that within the broader domain of cloud-based infrastructure, the resource allocation framework should be scalable across multiple distributed machines.

To this end, we note the following drawbacks in the prior art: (i) absence of service reliability constraints, (ii) absence of latency constraints, and (iii) lack of consideration for distributed cloud infrastructure. With a view to overcome the aforementioned shortcomings, in this chapter, we introduce a resource allocation framework for Cloud Brokers called Adaptive Containerization for Microservices in Distributed Cloud Systems (ACMDC) which helps reduce operating costs while ensuring a minimum guaranteed level of service, i.e., the SLA. It is anticipated that through the use of ACMDC, Cloud Brokers will able to streamline resource usage and costs, in turn providing Cloud Consumers with a greater degree of flexibility in their choice of cloud infrastructure.

To the best of our knowledge, this work is the first to present an SLA-based resource allocation framework for containerized microservices that takes into account both service latency as well as reliability. The rest of this chapter is organized as follows. Section 5.2 introduces the system model, the problem formulation, and our approach to the solution. Section 5.3 presents a comprehensive performance evaluation based on metrics relating to resource utilization, costs, and reliability, while Section 5.4 concludes the chapter.

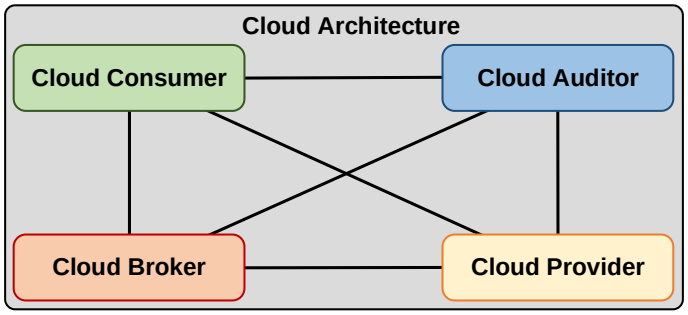


Figure 5.1: The cloud architecture under consideration [130].

5.2 System Model

In this section, first, we provide the necessary background on the cloud architecture under which ACMDC operates, along with the primary stakeholders in our system. Then, we provide the detailed problem formulation that mathematically characterizes the entities under consideration along with the objective and constraints. Finally, we present our approach to solving the resource allocation problem that has been posed.

5.2.1 Preliminaries

ACMDC operates within the realm of the general cloud computing architecture [130] shown in Figure 5.1. The four primary stakeholders within the cloud domain are the:

- **Cloud Consumer:** The entity that seeks to deploy network applications over cloud infrastructure owned by Cloud Providers. Within the context of this chapter, the communications service provider is the Cloud Consumer.
- **Cloud Provider:** The entity that owns the hardware infrastructure used to deploy services.
- **Cloud Broker:** The entity that negotiates the relationship between Cloud Consumers and Cloud Providers.
- **Cloud Auditor:** The entity that assesses and benchmarks the performance of Cloud Providers.

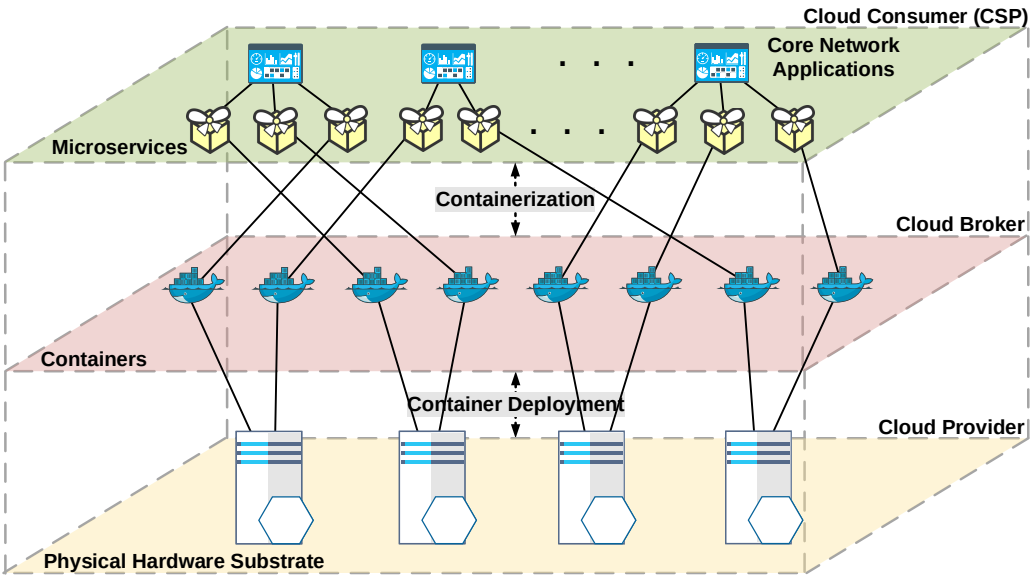


Figure 5.2: Containerized approach to microservices.

In what is known as the Infrastructure-as-a-Service (IaaS) model, the Cloud Consumer contracts with the Cloud Broker to deploy the corresponding core network applications in the cloud. The Cloud Broker typically leases cloud infrastructure from Cloud Providers in a wholesale manner. Therefore, the Cloud Broker is responsible for the containerization of each application's constituent microservices, and for the deployment of these containers onto the leased infrastructure.

5.2.2 Problem Formulation

As shown in Figure 5.2, the physical hardware substrate represents the physical machines leased by the Cloud Broker. A physical machine p is characterized by its computing resource capacity ($r(p)$), cost per unit of computing resource ($c(p)$) and operational cost ($m(p)$) incurred by the broker, link latency to the broker ($l(p)$), and probability of failure ($f(p)$). In other words, a physical machine p is represented by the tuple $\{r(p), c(p), m(p), l(p), f(p)\}$. Furthermore, a binary variable $o(p)$ is used to indicate the operational status of machine p,

such that,

$$o_p = \begin{cases} 1, \text{ if at least one container is deployed on } p; \\ 0, \qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{5.1}$$

The set of physical machines, P, serves as the hardware substrate for container deployment. Each core network application a is composed of a set of loosely coupled microservices which perform a specific set of tasks, i.e., the microservices that form the application are all independent of each other, and exchange messages through a message broker deployed at the Cloud Broker's premises.

In decomposing a network application into a set of microservices, we introduce the following quantities, $w(a, u)$ which denotes the number of requests that microservice u must fulfill for application a, $t(a, u)$ which denotes the time required by microservice u per unit computing resource to fulfill a request for application a, and $T(a, u)$ which denotes the execution deadline for all requests served by microservice u for application a. Furthermore, as part of the SLA requirement, a Cloud Consumer imposes a service latency requirement, $S(a, u)$, on the Cloud Broker, for every microservice u that serves application a, along with a cost penalty $P(a, u)$ that the broker must bear for every lost request of application a that is served by microservice u. Finally, we note that containerization is done on a per-request basis, i.e., for every request a new container is instantiated and deployed on the hardware substrate. This approach ensures maximum resiliency and allows for quick scaling of resources.

Next, the minimum amount of computing resource at physical machine p, $r(p, a, u)$, required to meet the execution deadline $T(a, u)$ is given by

$$r(p, a, u) = \frac{x(p, a, u)t(a, u)}{T(a, u)} \quad \forall\, p \in P,\ a \in A,\ u \in U, \tag{5.2}$$

where $x(p, a, u)$ is the number of containers deployed on machine p of microservice u that serve application a. Since a physical machine can only support a limited number of

containers, we enforce a machine capacity constraint as follows

$$\sum_{a \in A} \sum_{u \in U} r(p, a, u) \leq r(p) \qquad \forall\, p \in P. \tag{5.3}$$

Furthermore, to ensure that the total number of containers does not exceed the total number of requests, we have

$$\sum_{p \in P} x(p, a, u) = w(a, u) \qquad \forall\, a \in A,\ u \in U, \tag{5.4}$$

and the relationship between the indicator variable $o(p)$ and decision variable $x(p, a, u)$ is given by

$$\sum_{a \in A} \sum_{u \in U} x(p, a, u) \leq M o(p) \qquad \forall\, p \in P, \tag{5.5}$$

where M is a large positive real number. As mentioned previously, the framework presented herein also takes into account the SLA between the Cloud Consumer and the Cloud Broker, accordingly, in order to meet the service latency requirement, we introduce the following constraint

$$T(a, u) + \sum_{p \in P} x(p, a, u) l(p) \leq S(a, u) \qquad \forall\, a \in A,\ u \in U. \tag{5.6}$$

(5.6) takes into account both the execution deadline as well as the time it takes to communicate the microservice output back to the message broker. The second SLA metric we take into consideration is system reliability. Since the probability of machine failure, $f(p)$ is a Bernoulli random variable, the expected number of failed (or lost) requests at machine p is given by $\sum_{a \in A} \sum_{u \in U} x(p, a, u) f(p)$, and the expected value of the associated overall cost penalty can be expressed as

$$v(p) = \sum_{a \in A} \sum_{u \in U} x(p, a, u) P(a, u) f(p) \qquad \forall\, p \in P. \tag{5.7}$$

On other hand, the operating cost and resource cost per machine p can be expressed as

$$u(p) = m(p)o(p) + \sum_{a \in A} \sum_{u \in U} r(p, a, u)c(p) \qquad \forall\, p \in P. \tag{5.8}$$

Finally, the optimization problem (OP) associated with ACMDC is given as follows

$$\begin{aligned}
(OP) \quad \min_{x(p,a,u),\ o(p)} \quad & \sum_{p \in P} \left(w_u u(p) + v_u v(p)\right) \\
\text{s.t.} \quad & (5.3) - (5.6), \\
& o(p) \in \{0, 1\} \qquad \forall\, p \in P, \\
& x(p, a, u) \in \mathbb{Z}^+ \qquad \forall\, p \in P,\ a \in A,\ u \in U,
\end{aligned}$$

with $w_u + w_v = 1$, $0 \leq w_u \leq 1$, and $0 \leq w_v \leq 1$.

5.2.3 System Operation

ACMDC follows a reactive event-driven model which allows us to maintain high elasticity with low complexity. Examples of such events include the arrival of a new network application request at the Cloud Broker, a change in the parameters of an existing application, and a change in the parameters of the underlying hardware substrate. Accordingly, there are two major event classes– new application events and change events. As shown in Algorithm 2, on receiving an event notification, the Cloud Broker first checks the event type.

For new application events, OP is solved twice, first under the assumption that the existing applications cannot be migrated, and a second time with the aforementioned assumption relaxed. Clearly, the second approach is optimal but it entails a migration cost associated with moving containers across physical machines. However, if the cost difference between the two does not exceed the migration cost, the Cloud Broker utilizes the solution offered by the first approach and does not perform migration. On the other hand, for change events, the Cloud Broker first checks the validity of the existing container deployment, if the existing

deployment is no longer feasible, OP is solved with the new parameters, and the Cloud Broker performs container migration. While if the existing deployment is deemed feasible, migration is only done if the cost difference exceeds the migration cost. In this manner, the system is able to seamlessly adapt to different events.

Algorithm 2 Adaptive Containerization for Microservices in Distributed Cloud Systems (ACMDC)

```
initialization: process existing event batch
while new event do
    check event type
    if new application event then
        solve OP with migration disabled
        solve OP with migration enabled
        check difference in cost outlay
        if cost difference > migration cost then
            accept solution with migration enabled
            perform migration
        else
            accept solution with migration disabled
        end if
    else
        check feasibility of existing container deployment
        if existing deployment not feasible then
            solve OP with migration enabled
            accept solution with migration enabled
            perform migration
        else
            solve OP with migration enabled
            check difference in cost between existing and proposed deployment
            if cost difference > migration cost then
                accept solution with migration enabled
                perform migration
            else
                maintain existing deployment
            end if
        end if
    end if
end while
```

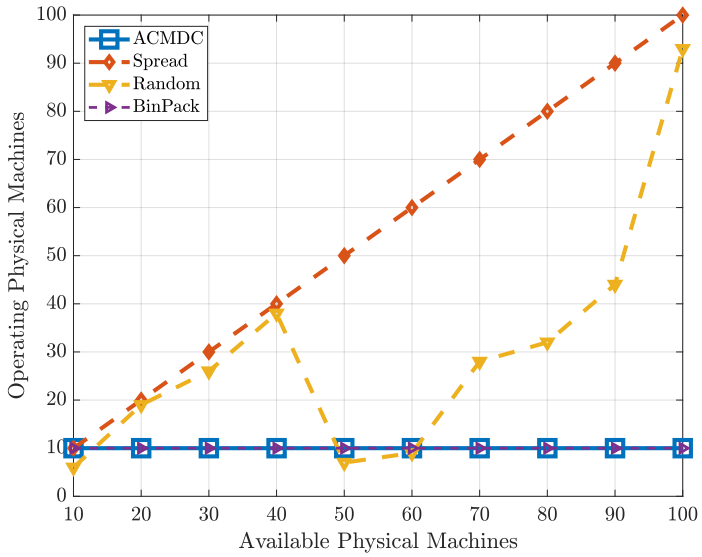


Figure 5.3: Number of active machines relative to the total available machines.

5.3 Performance Evaluation

In this section, we evaluate the performance of the ACMDC framework by considering a real-world cloud deployment scenario and comparing the results obtained against the deployment strategies Spread, Random, and BinPack used by some of the most popular container orchestration platforms today [131]. More specifically, concerning these three strategies, we note the following:

- **Spread:** Under the Spread strategy, the physical machine with the least number of active containers is chosen for container deployment.
- **Random:** The Random strategy does not seek to optimize for any performance metric, instead, machines are chosen at random for container deployment.
- **BinPack:** BinPack follows the classical bin packing approach [132, §18], wherein the system prioritizes physical machines with the most number of existing containers.

The first performance metric we take into consideration is the number of active machines vs. the total number of physical machines as shown in Figure 5.3. Figure 5.3 characterizes the system response to an increasing number of nodes in the physical substrate. For a

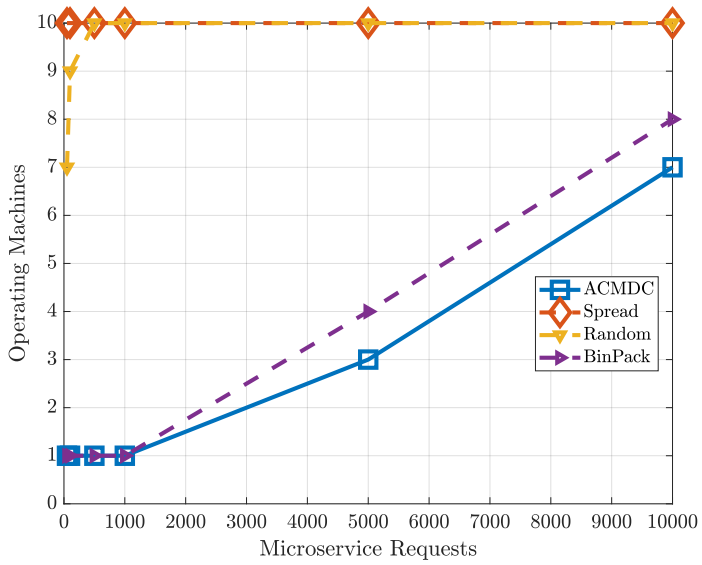


Figure 5.4: Number of active machines relative to the number of requests.

fixed set of $10,000$ requests, we see that both the Spread and Random strategies use up an increasing number of machines which is sub-optimal from an operating cost perspective, on the other hand, ACMDC and BinPack both maintain a constant 10 active machines. This result can be attributed to the fact that 10 physical machines are sufficient to meet the requirements of the $10,000$ requests under consideration while achieving minimal cost outlay.

Figure 5.4 represents the increase in the number of active machines as the total number of requests increases from 50 to $10,000$, for a substrate of 10 machines. As expected, Spread makes use of all 10 machines, Random eventually converges to 10 machines too, as the volume of requests increases. Furthermore, while both ACMDC and BinPack utilize fewer than 10 machines, ACMDC edges out BinPack by requiring a fewer number of active machines to serve a maximum of $10,000$ requests. For example, while BinPack requires 8 machines to serve $10,000$ requests, ACMDC is able to serve the same number of requests with 7 physical systems.

In a similar vein, Figure 5.5 characterizes the normalized cost outlay associated with the containerization and deployment of microservices, and further demonstrates a major advantage offered by ACMDC. The costs shown in this figure have been obtained by increasing

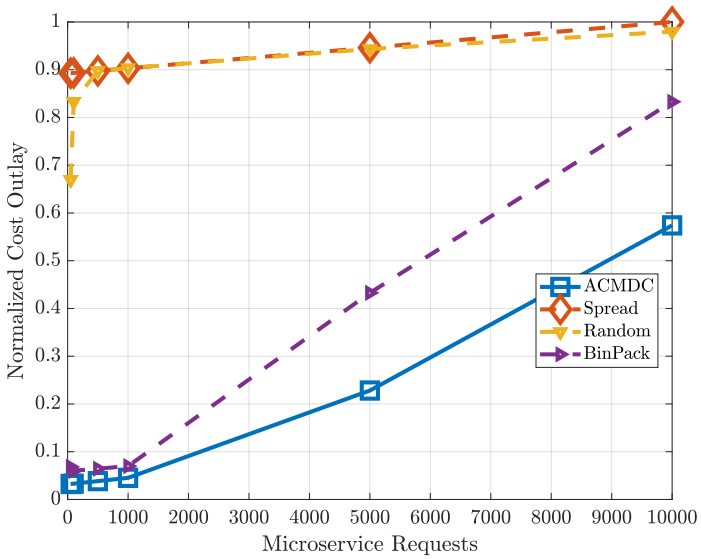


Figure 5.5: Normalized cost outlay relative to the number of requests.

the number of microservice requests from 50 to $10,000$, and then normalizing the result by the highest cost value, which happens to be achieved by the Spread strategy. From the figure we see that, as the number of requests scales to $10,000$, ACMDC is able to achieve cost savings of over 40% compared to Spread and Random, and nearly 35% compared to BinPack. The results thus obtained further reinforce the superiority of ACMDC which achieves the lowest cost outlay.

Finally, system reliability has been characterized in Figure 5.6, which represents the failure probability of the requests. While the service latency requirement is enforced as a constraint, system reliability forms a part of the objective. Ideally, we would like to place a majority of the containers on machines with a low probability of failure $f(p)$, however the resource cost $c(p)$ and operating $o(p)$ cost serve as impediments to this idealized deployment. In other words, since system reliability is characterized by a cost penalty and the problem objective deals with minimizing the overall system cost, we are willing to accept a slightly less reliable system in exchange for a lower overall cost. For example, as shown in Figure 5.6, we see that ACMDC favors machines with $f(p) = 0.2$, over those with $f(p) = 0.1$, however, on further inspection of the substrate parameters, we find that this result is the direct outcome of the higher per unit resource cost associated with more reliable machines. At the same

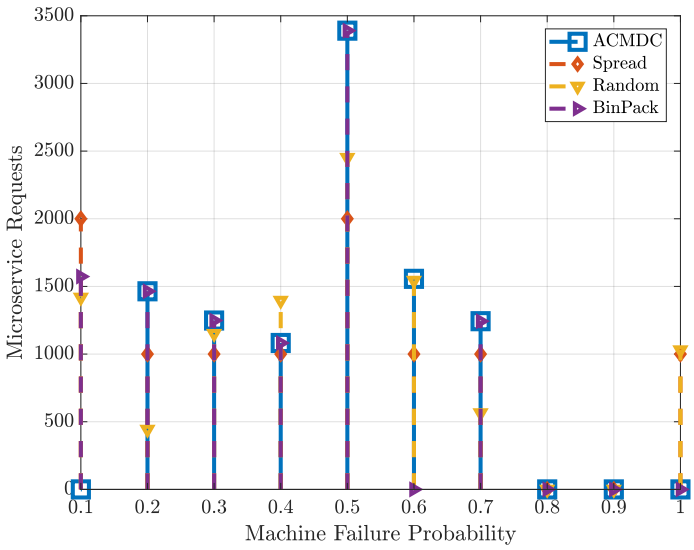


Figure 5.6: Number of requests relative to the machine failure probability.

time, we note that while the performance evaluation in this section accords equal importance to $u(p)$ and $v(p)$, a Cloud Broker is free to adjust the weights w_u and w_v based on their preferences.

5.4 Highlights

In this chapter, we have presented an optimal containerization framework for the cloud-based deployment of core network microservices, ACMDC. ACMDC helps Cloud Brokers minimize their total cost outlay while meeting the SLA requirements set forth by Cloud Consumers, i.e., the communications service providers. As part of ACMDC, we have developed mathematical representations for the physical substrate, network applications, and microservices, and introduced constraints that accurately reflect the cloud domain. Furthermore, ACMDC has been presented as a reactive event-driven framework that enables elasticity with low complexity. Finally, we have validated the superior performance of ACMDC by comparing it to the state-of-the-art deployment strategies based on metrics relating to the number of active physical machines, the total cost outlay, and system reliability.

CHAPTER 6

END-TO-END SYSTEM DESIGN FOR THE INTERNET OF SPACE THINGS

Up until this point, the thesis has primarily focused on cellular communications covering both the RAN and core domains. However, global network coverage is a vital component of wireless ubiquity, and therefore, satellite networks are equally important for realizing our vision. To this end, starting with this chapter, the focus of the thesis will now shift towards satellite systems.

6.1 Motivation

Over the past four years, the global Internet of Things (IoT) market has witnessed rapid growth, expanding by over 30% in the past year alone [133]. As it stands, worldwide expenditure on IoT solutions is expected to exceed 7 trillion by 2020 [134]. Thus, it is no longer a question of if or when, but rather of just how deeply IoT will transform the industry going forward. While smart devices, enhanced connectivity, and emerging standards have played a vital role in the ever increasing adoption of IoT, global connectivity remains a key concern [134]. For example, a freight transportation company would prefer having ubiquitous access to its consignments and thus would require a connectivity solution that scales across both urban and rural areas, as well as oceans.

However, the majority of IoT solutions today rely on the heterogeneous integration of wireless personal area networks (WPANs, e.g., Bluetooth, Zigbee, Z-Wave, among others), Wireless Local Area Networks (WLANs, e.g., WiFi), Wide Area Networks (WAN, i.e., 3G, 4G, and 5G), and, more recently, Low Power WANs (LPWANs, e.g., LoRa, Sigfox, and NB-IoT). The primary drawback of the aforementioned connectivity solutions is their reliance on pre-existing infrastructure. In addition to the freight transportation example above, a myriad of global connectivity use cases are charcterized by deployments in geographical

areas which are either difficult to provide coverage to, such as the North and South Poles, or in which the cost of installation outnumbers the potential benefits due to low-density of population or high-cost of infrastructure such as remote forests and deserts. To this end, satellites have long been used as a means for providing global coverage, from Iridium, Teledesic, Globalstar, Celestri, and SkyBridge [135] in the 1990s, to the more recent concept of Internet of Space (IoS) which relies on Low Earth Orbit (LEO) satellites [136].

Nevertheless, many of the factors that caused Teledesic and Celestri to fail are still prevalent today. More specifically, in the context of a fast-changing IoT landscape, we note that traditional satellites suffer from certain characteristic drawbacks:

- **Long development cycles:** They have long development timelines ranging from three years for commercial ventures to over seven years for government programs [137].

- **High costs:** They often have very high costs associated with the development, construction, and launch phases resulting in high barriers to entry for new operators and vendors. Consequently, the development of satellites has been restricted to a few major players. For example, it is projected that the Iridium NEXT system will cost over $3 billion [138] to develop and deploy.

- **Increasing congestion:** They rely on the traditional frequency bands that are becoming increasingly prone to congestion [139]. However, due to the high cost and long development timelines, a traditional LEO constellation is limited in terms of the number of satellites it contains [140], thus precluding the use of higher frequencies, particularly those in the THz range due to the resulting large separation between adjacent satellites.

- **Lack of sequential redundancy:** Traditional satellites do not make use of sequential redundancy [141]. In the traditional approach, a new development program is started only after the development, build, and launch stages of the previous program have been completed, making it difficult to adapt to the needs of a rapidly changing market.

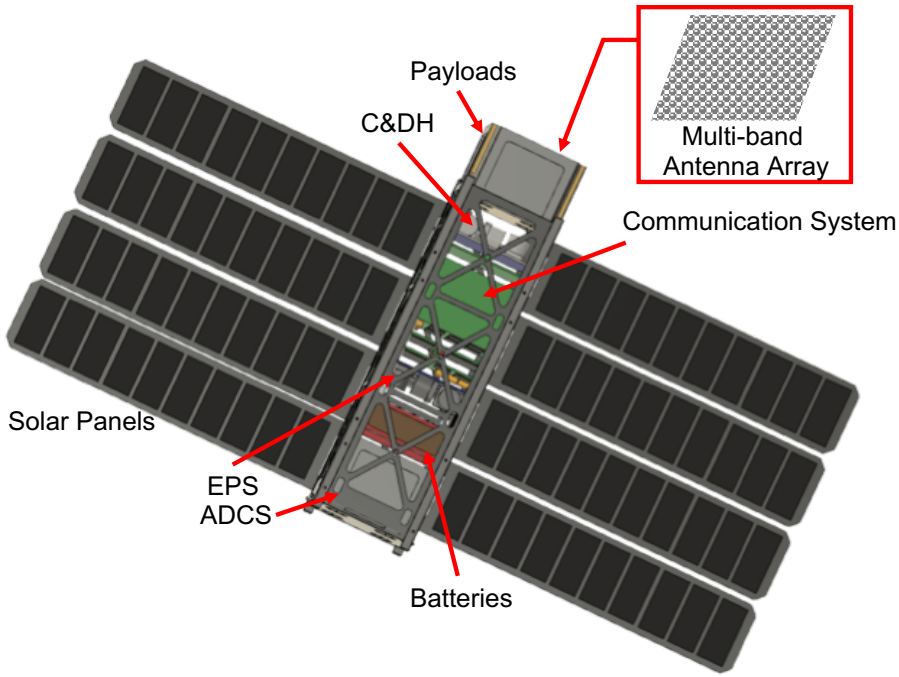


Figure 6.1: Preliminary design of our next-generation 3U CubeSat [143].

- **High-risk exposure:** Failure at any stage leads to huge setbacks in terms of cost and time. For example, the failure of the Hitomi Telescope in 2016 led to a loss of over $\$200$ million, and 10 years of research [142].

The aforementioned drawbacks have motivated a sea change in the satellite infrastructure landscape which has witnessed the emergence of a new class of miniaturized satellites known as CubeSats. Originally envisioned for university education and research purposes, CubeSats are seen as a promising solution to realize global satellite networks at much lower costs. In particular, while even little LEOs weighing under 50 kg cost several tens of millions of USD, the total cost associated with an advanced CubeSat is typically less than a million [55]. In addition, the short time frame from development to deployment makes CubeSats an efficient deployment option.

CubeSats have uniform cubic sizes denoted as 1U, 2U, and so on, where "1U" refers to a $10 \times 10 \times 10$ cm^3 cube, and can be used for applications in numerous research fields including biochemistry, astrophysics, and telecommunications [144]. With a view to further enhance the applicability of CubeSats to different use cases, we have recently introduced

a next-generation 3U CubeSat hardware design, as shown in Figure 6.1 that is able to support multi-band wireless communication at microwaves, mm-wave, THz, and optical frequencies [145].

In this chapter, we introduce the new concept of the Internet of Space Things (IoST). In IoST, CubeSats not only play the role of network infrastructure providing globally scalable connectivity, but also function as passive and active sensors of the physical world. More importantly, the closed-loop integration of CubeSat sensing and CubeSat communications results in a new cyber physical system with innovative applications spanning ground, air, and space. We further note that while IoST is patently different from the IoS systems that have preceeded it, we envision leveraging additional connectivity provided by LEOs, MEOs, and GEOs as required. We propose the use of software-defined networking (SDN) and network function virtualization (NFV) as means to dramatically improve network resource utilization, simplify network management, and reduce operating costs. The SDN and NFV-based system architecture allows for a dynamic and scalable network configuration, integrated service delivery, logically centralized network control, and enables the IoST-as-a-Service (IaaS) paradigm.

The major contributions and novelties of this chapter are summarized as follows:

1. A targeted set of applications that will benefit from IoST. We show how IoST can serve a variety of different use cases and the different roles the CubeSats play in each.

2. Complete system architecture with component-level description. We describe in detail the different components of the IoST architecture and their interactions.

3. Solutions for tackling peculiarities of the space environment. We introduce the new concepts of stateful segment routing (SSR) and virtual CSI (vCSI) that are geared towards overcoming challenges such as long delays, and temporal topological variations that are associated with operating in the space environment.

4. The primary research challenges involved. We present the key challenges that serve

as hurdles to the practical realization of IoST, and motivate possible solutions that are expected to guide research in this domain.

5. Preliminary system performance evaluation. We provide an initial insight into the potential of IoST, and lay the groundwork for a more rigorous analytical model of the system.

The remainder of this chapter is organized as follows. First, in Section 6.2, we present the use cases of IoST, including utilizing IoST as a remote monitoring system, as a pervasive backhaul, and, ultimately, as an integrated cyber physical system. Then, in Sections 6.3–6.6, we present the system architecture, and provide an in-depth description of the different system components. In Section 6.7, we present system procedures that are geared towards making efficient use of the space environment, along with the challenges associated with each procedure, followed by the system performance evaluation in Section 6.8. Finally, we conclude this chapter in Section 6.9.

6.2 Application Scenarios

IoST is envisioned as the enabling technology for a variety of new transformative applications beyond traditional IoT. CubeSats in IoST are not a mere backhaul network providing wireless connectivity, but also engage in active and passive sensing themselves. As shown in Figure 6.2, the physical architecture of IoST consists of the IoST Hubs, on-Earth and near-Earth sensing devices, and the CubeSat network. The IoST Hubs communicate with the CubeSats and house a large portion of control framework for the entire network, whereas the CubeSats operate in the exosphere (altitudes of 600 km and above) forming the network in space to receive, transmit, and relay data efficiently. Also shown in Figure 6.2 is the customer premises, which in the context of IoST, serves as the termination point or destination for the data. More specifically, as shown Figure 6.2, while passive sensing provides monitoring and reconnaissance capabilities, the IoST Hubs and active sensing applications

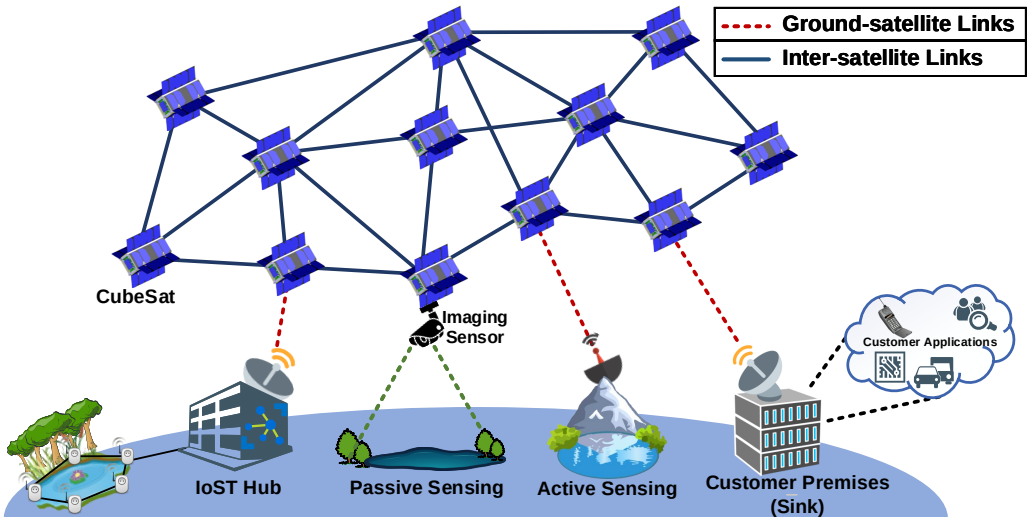


Figure 6.2: System overview of the Internet of Space Things with CubeSats.

utilize the CubeSat network as a backhaul. Together, they work in tandem to achieve a truly integrated cyber-physical system. In more general terms, the application scenarios of IoST can be divided into three categories based on functionality: (i) monitoring and reconnaissance, (ii) in-space backhaul, and (iii) cyber-physical integration. In this section, we provide detailed descriptions about the use cases of IoST and discuss how IoST can solve the challenges faced by terrestrial wireless networks.

6.2.1 Monitoring and Reconnaissance

CubeSats have an extensive role to play in aerial reconnaissance and monitoring. The use of imaging sensors, such as multi-resolution cameras that are able to capture infra-red, visible, and ultra-violet images is central to applications relating to terrain monitoring, and disaster prevention and monitoring, to name a few [54]. More specifically, in areas susceptible to earthquakes, such as in California and Nevada in the U.S., buildings, bridges, tower cranes, and other highly-elevated constructions need to be monitored to ensure their stability, as shown in Figure 6.3. Apart from their use in terrain monitoring and disaster prevention, the CubeSats in IoST can also be equipped with sensors for environmental monitoring. For example, they could be dropped out of a helicopter or drone into a growing oil spill, or onto

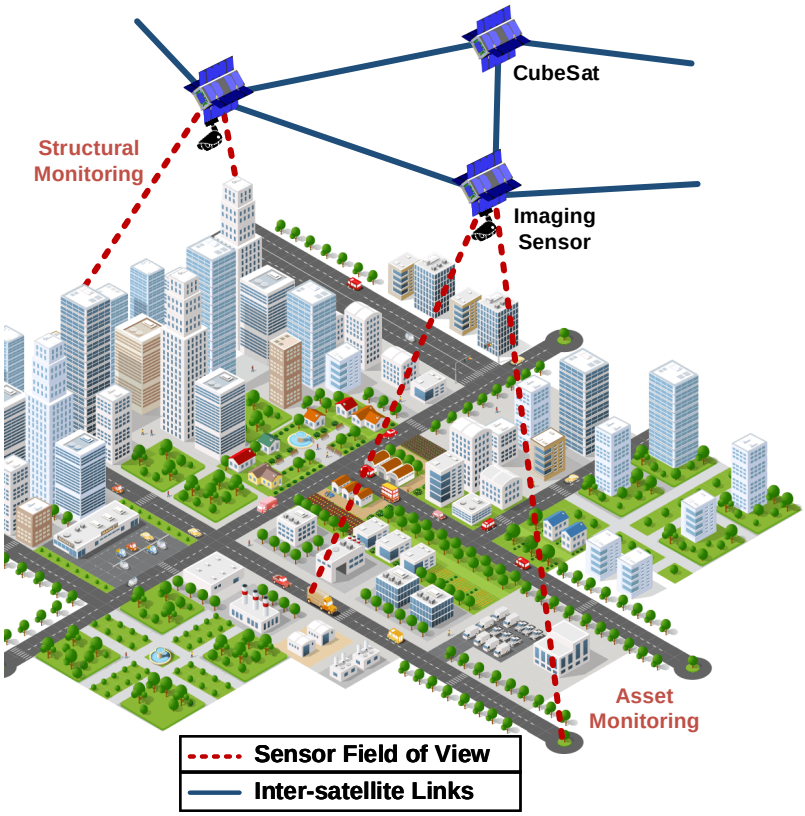


Figure 6.3: Monitoring and reconnaissance applications.

an active volcano to track lava flows.

6.2.2 In-Space Backhaul

In this category, as shown in Figure 6.4, IoST provides an in-space backhaul for data reporting and forwarding between CubeSats and other ground infrastructure. While terrestrial networks cover urban areas well, in remote areas where user density is much lower, most cellular carriers and Internet service providers largely disregard investing in infrastructure construction and maintenance. In IoST, connections are realized through CubeSat networks, which do not necessarily require ground infrastructure, thus saving both construction costs as well as helping preserve the natural landscape. While some existing solutions also provide satellite backhaul connectivity, higher reconfigurability and scalability have not been considered and implemented thus far. Thanks to the multi-band connectivity of our next-generation 3U CubeSats, high-speed reliable service can ensure minimal latency for the

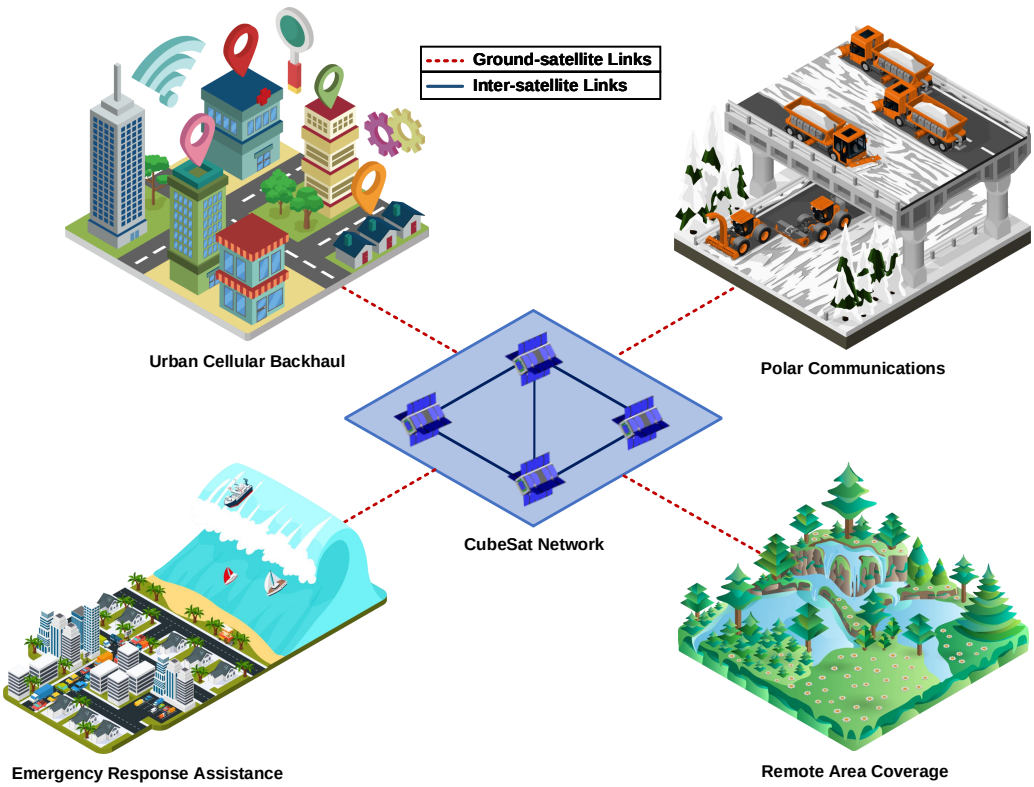


Figure 6.4: In-space backhaul scenarios.

in-space backhaul network.

Additionally, new M2M applications in remote areas can also rely on IoST as a backhaul in space to track and manage just-in-time inventories, enterprise fleets, energy grids, remote infrastructure, emergency operations, personnel deployments, and natural processes. For example, in the North and South Poles, where currently satellite coverage is intermittent, the scientific, operational, and weather data collected from measurement equipment can be transferred to CubeSats in IoST and forwarded to data processing centers in a timely manner. IoST can also send commands from control centers to the equipment to change operation modes or perform necessary measurements.

Moreover, the connectivity of IoST can also be leveraged in non-remote areas where emergency connections are needed, such as in areas affected by earthquakes, tsunamis, or tornadoes where ground infrastructure might be subject to damage. Furthermore, in the context of network security, wherein terrestrial infrastructure is vulnerable to compromise,

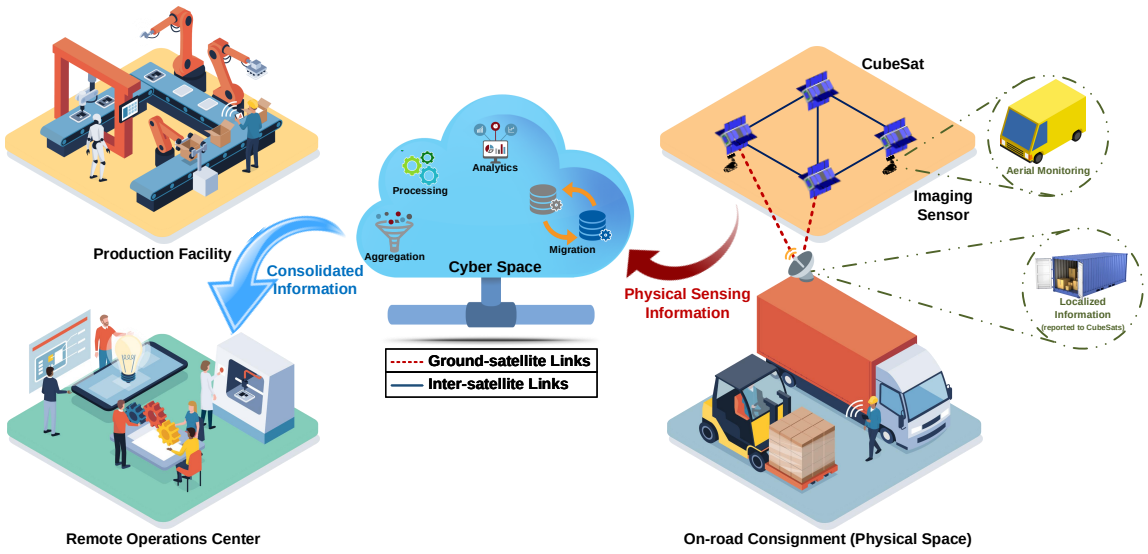


Figure 6.5: Cyber-physical integration through IoST.

the remote and flexible configuration of IoST can serve as a secure secondary backhaul.

6.2.3 Cyber-Physical Integration

Currently, the sensor networks on Earth and in space are isolated in terms of sensing and data collection, among other functionalities. Nevertheless, a fully integrated system should coherently combine and utilize both local and remote sensors. For example, in autonomous driving, satellite imaging combined with local in-vehicle sensors can achieve optimal global routing and traffic monitoring.

In addition to intelligent transport in urban areas, cargo transportation systems across both land and sea routes can exchange data with CubeSats. More specifically, utilizing the benefit of our low-orbit CubeSat network, IoST can keep track of all in-transit consignments by requesting data from local sensors in trucks and then forwarding to the freight carriers' database, as shown in Figure 6.5. Meanwhile, the sensors in CubeSats can also provide information about road and weather conditions to the truck drivers.

Moreover, the integrated service can also enhance the performance of unmanned aircraft systems (UAS), such as drones and balloons, in applications including Internet access for underdeveloped areas, aerial photography, product deliveries, surveillance, and so on. IoST

can coordinate these drone swarms in order to improve the overall network performance. For example, for drones operated by different owners in the same area, it is important to have a coordinator to help them "see" each other to avoid collisions. The controllers on the ground have limited sights because of obstructions, whereas the IoST CubeSats with their more global field of view (FOV) are better suited to assist in coordination tasks. Specifically, such coordination tasks involve CubeSats' passive sensing of aircrafts in their FOV, active reporting of localization information to each other, and instructing altitude data to drones as well as ground infrastructure.

Thus far, we have discussed in great detail the use cases of IoST in the context of on or near-Earth scenarios. However, as has been demonstrated by NASA recently [146], CubeSats can also be designed for deep space applications. In the context of deep space applications, IoST can be used for interplanetary data relaying, sensing and monitoring on asteroids, Mars, and the Moon, as well as even farther in to the depths of space. We posit that promising CubeSat missions will be enabled with future advancements in physics, electronics, and telecommunications. In this section, we have largely focused on the current on-Earth and near-Earth explorations of IoST.

6.3 System Architecture Design

Up until this point we have discussed the use cases that motivate the need for IoST. Next, we focus on the control and operation of IoST, for which we advocate for the use of an SDN and NFV-based approach. In this section, we detail the SDN and NFV-based system architecture. First, we discuss the key system features in Sections 6.3.1, followed by the architectural overview in Section 6.3.2. Then, we delve into each of the architectural layers in Section 6.4, 6.5, and 6.6.

6.3.1 Key Features

We have already seen how the use of SDN and NFV can simplify network management. More specifically, in the context of satellite networks, the ground stations and satellites follow highly vendor specific implementations, and consequently the "closed" nature of hardware and software hampers network evolution. Furthermore, we note that in order to realize a truly cyber-physical system, the CubeSat transport network must be tightly integrated with the different application scenarios; considering a remote monitoring example, this integration could take the form of joint optimal routing over both the sensor field and the satellite network, which is not possible in traditional network setups.

Advances in satellite infrastructure design, and the aforementioned CubeSat concept also serve as key drivers in enhancing the feasibility of SDN in the space segment. The availability of high quality off-the-shelf components in recent years, and relatively low-cost launch facilities has seen the number of CubeSat missions surge rapidly, exceeding 150 in 2017 [147]. Furthermore, the computing hardware of CubeSats can be built completely from commercial off-the-shelf (COTS) components [148]. The use of CubeSats also allows infrastructure providers to significantly lower their deployment and operating costs in comparison to traditional satellite networks [149]. Similar to white-box switching hardware common in wired SDN/NFV networks, the use of COTS hardware, limited onboard processing capability, and low costs make CubeSats a perfect fit for the data forwarding component of the network, i.e., the data plane.

Through the use of SDN and NFV, IoST incorporates the following key features:

- **Network scalability:** The logical centralization of control capabilities offered by SDN allows for intelligent routing of massive amounts of traffic by ensuring optimal utilization of network infrastructure thereby meeting the strict quality of service (QoS) requirements of a variety of applications. The applications can also exercise control over network policy through their interaction with an abstracted version of the underlying infrastructure. The

centralized control of data flow offered by SDN/NFV further enables and simplifies data aggregation.

- **IoST-as-a-Service (IaaS):** Emerging and growing applications, such as environment monitoring, smart grid, smart city, smart transportation, e-health, smart home, and remote sensing require highly differentiated networking capabilities to be integrated and deployed over the same network infrastructure. The network virtualizability of IoST allows CubeSats and other sensing devices to be treated as a service rather than as a physical asset. The infrastructure operator and service provider now become two different entities. IoST provides service providers with the ability to control, optimize, and customize the underlying infrastructure without owning it nor interfering with the operations and performance of other tenants, thus leading to cost-efficient operations and enhanced QoS. Such multitenancy is implemented through network slicing that ensures resource isolation between tenants.

- **Ubiquitous connectivity:** IoST is expected to bridge diverse technologies to enable new applications by connecting physical objects, vehicles, appliances, devices, buildings, and so on to a diverse set of endpoints both on as well as near-Earth. IoST by design includes support for different environments namely the terrestrial, underwater, and underground domains, augmenting endpoints in each with satellite transport.

- **Network security:** Since SDN and NFV centralize the network management and control, they will be able to handle the network security at different layers efficiently. Furthermore, as SDN and NFV profile each flow differently, they can manage the traffic from various services which have different security profiles. This fact would ease the additional processing of less secure applications. IoST implements an identity-based authentication scheme, and robust access policy framework for protecting the Control and Management Layer from unauthorized access, and establishing trust primitives. Security is built into the IoST architecture, as well as delivered as a service to protect the availability, integrity,

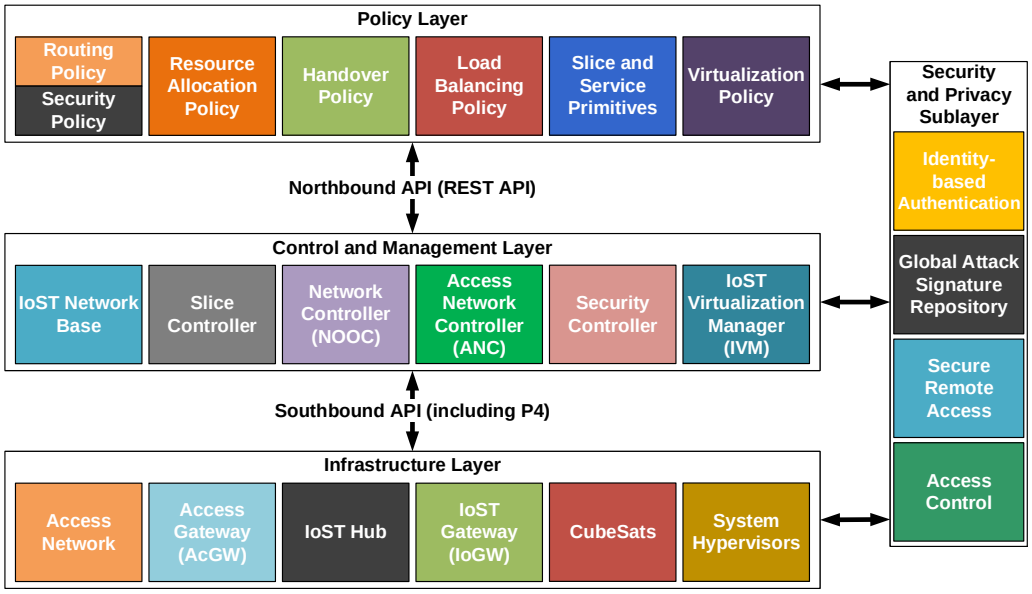


Figure 6.6: The IoST system architecture layers.

and privacy of all connected resources and information.

6.3.2 System Architecture Overview

In order to guarantee control and data plane separation, the IoST architecture follows a layered structure as shown in Figure 6.6 consisting of the following:

- **Infrastructure Layer:** It represents the physical hardware in the network such as sensing devices, switches, gateways, servers, and CubeSats, along with hardware virtualization solutions.

- **Control and Management Layer (CML):** It is responsible for overall network orchestration, operations, and management. In SDN and NFV parlance, the CML is analogous to the control plane, and management and orchestration entity. The CML includes a security and privacy sub-layer that handles network security.

- **Policy Layer:** It allows external entities to interact with the IoST system. It provides tenants with a way to push their network policies, as well monitor as network status.

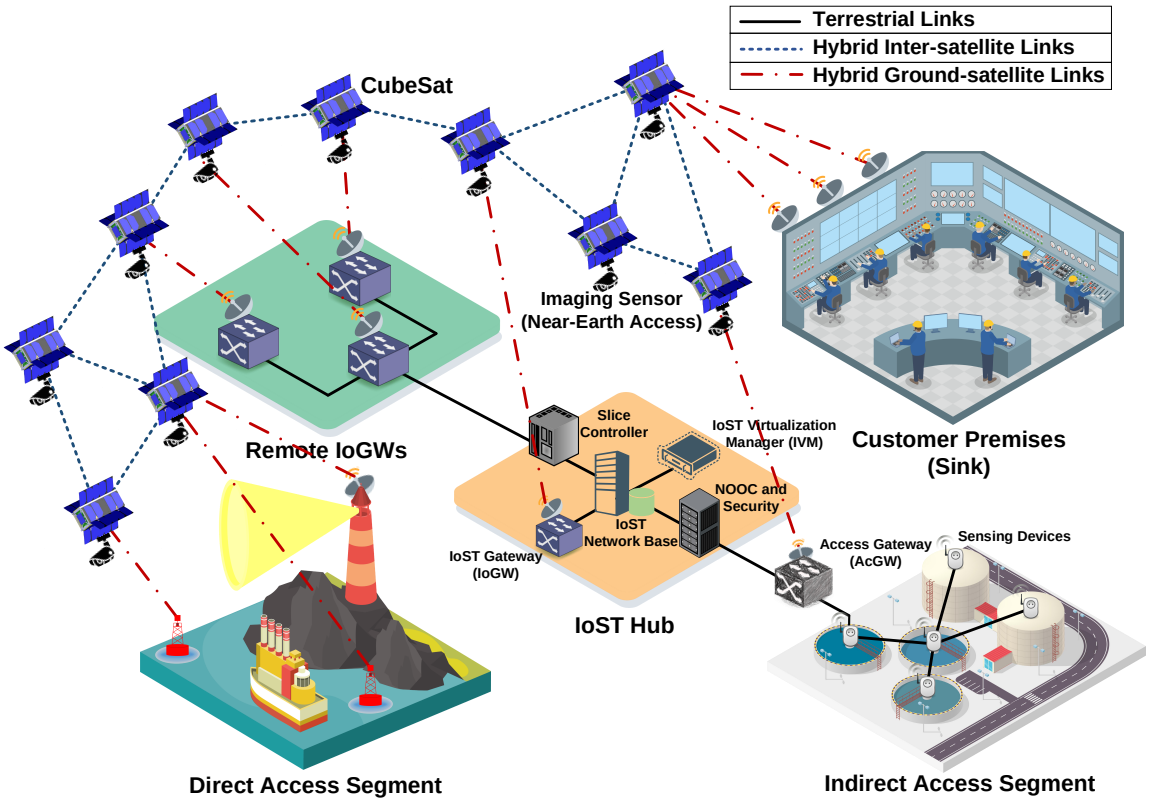


Figure 6.7: The IoST system architecture design.

Together, these layers enable a dynamic and scalable network infrastructure, resource virtualization, the integration of heterogeneous services and technologies, and the provisioning of security and privacy. The Policy Layer interacts with the CML through a RESTful NBI, while the SBI between the CML and infrastructure layer makes use of the protocol independent P4 language [64] where feasible. Figure 6.7 represents the overall view of the system showing both terrestrial, as well as near-Earth applications. In the following sections, we examine each of these layers in detail.

6.4 Infrastructure Layer

As shown in Figure 6.7, the Infrastructure Layer forms the underlying physical fabric of the system. It consists of the sensing devices, switches, gateways, and servers that form the Access Network and IoST Hub, in addition to the CubeSats. The Infrastructure Layer

also consists of hypervisors operating under the umbrella of a IoST Virtualization Manager (IVM) which are responsible for hardware virtualization that makes network function deployment and management seamless. In the following, we provide a component-level description of the Infrastructure Layer.

6.4.1 Access Network

The Access Network is characterized by the presence of a virtual sensor network [150] which serves as the data source. Within the context of IoST, sensing devices can belong to one of three categories: (i) the direct access segment, (ii) the indirect access segment, and (iii) the near-Earth segment. As the name suggests, sensing devices that can communicate with CubeSats directly form the direct access segment. The direct access segment is best suited for applications wherein data transfer is intermittent, and the devices themselves are not power constrained. In the absence of additional ground infrastructure, one of the sensing devices itself functions as a cluster-head providing localized control for the sensor field. For example, in freight transportation systems, the sensors that monitor consignment temperatures do not need to transmit data continuously, and have a ready power source available from the vehicle. In addition to low data volume applications, the direct access segment is also perfect for use in areas where ground infrastructure deployment is not feasible.

On the other hand, energy constrained devices that are not suited for direct communication with the CubeSat network form the indirect access segment. The indirect access segment is characterized by the presence of an Access Gateway (AcGW) which aggregates the data, and either forwards it directly to the CubeSats or through the IoST Gateway (IoGW). The use of the IoGW is particularly useful when the IoGW is physically close to the AcGW, because: (i) additional delay is negligible due to physical proximity, and (ii) control traffic is minimized due to pre-processing of the traffic at the IoST Hub. The indirect access segment is well suited for underwater sensing applications, for example, wherein the

sensing devices are energy constrained, and recharging is difficult. In such scenarios, the buoys on the ocean surface can serve as AcGWs, forwarding data to the CubeSats.

Aerial reconnaissance and monitoring applications make extensive use of CubeSats augmented with sensing payloads, thus forming the near-Earth segment. The near-Earth segment is unique in that it makes use of CubeSats for both sensing as well as communication. The IoST architecture provides the flexibility of using different segments in tandem, providing a holistic data collection framework. The devices leave the decision-making to the CML by interacting with it through standardized interfaces, i.e., the SBI. For example, the data-forwarding function is controlled by a set of match and action tables. In this way, it is possible to set up customized actions on the different devices to implement firewall, packet classification, and load balancing functionalities in the data plane. Furthermore, the P4 operations– configure and populate are carried out by the CML, in tandem with the policy layer.

6.4.2 IoST Hubs

IoST Hubs represent the ground stations in IoST. To maintain robust connectivity with CubeSats, they are distributed globally across data centers. A typical Hub implementation includes the IoGW, along with network control infrastructure such as the IoST Network Base, the IoST Virtualization Manager (IVM), Slice Controller, the Network Operations and Orchestration Controller (NOOC), and the Security Controller, each of which are detailed in Section 6.5. Furthermore, an IoST Hub can have multiple distributed IoGWs attached to it, all of which may not be active simultaneously. In this manner, an IoST Hub serves as a point of interconnection between the terrestrial Access Network and the CubeSat constellation. By virtue of housing the control and management entities, the IoST Hubs also enable interactions with tenants and other stakeholders.

6.4.3 CubeSats

While we have previously presented the design aspects in detail [145], here we reinforce that CubeSats form the primary data forwarding component of the IoST system. Depending on the application, the CubeSats may perform only the forwarding function, or take part in both sensing and data collection. Owing to the limited processing capacity of these small satellites, they are devoid of control logic, instead receiving control directives from the CML. The IoST architecture incorporates a great deal of flexibility which allows the deployment to be tailored to a specific application, if required. Certain elements of the infrastructure layer depend on the nature of the application, i.e., for the indirect access and near-Earth segments, an AcGW is not required.

6.4.4 Resource Virtualization

We have already established that network slicing is central to the IoST framework. Accordingly, the virtualization of hardware resources forms a major aspect of the infrastructure layer. Within IoST, we identify three primary classes of hardware: (i) classical hardware– computing resources in servers and switches, (ii) virtual sensor networks– computing and radio resources in sensing devices, and (iii) CubeSat networks– computing and radio resources in CubeSats. Insofar as classical hardware is concerned, the virtualization of computing resources– processing, memory, and storage has been investigated a great deal with several virtualization solutions such as KVM [70], LXC [71], Xen [72], and Hyper-V [73] being readily available. Similarly, there exist network hypervisors that provide more than one networking context per physical networking device to allow for the provisioning of differentiated services– FlowVisor [74] and its extensions [75]; and the recently proposed HyPer4 [76] and HyperV [77] hypervisors for virtualizing P4-based switching hardware.

On the other hand, virtualization in the sensor space has been motivated by the proliferation of multi-function sensing devices, and the corresponding need for concurrent execution of multiple applications. To this end, RIOT OS [151] is a promising solution under active

development that brings real-time multi-threading support to resource constrained sensors, along with a full TCP/IP stack with 6LoWPAN support. While RIOT OS allows for multiple applications to run on top of a single sensing device, the virtualization of radio resources remains an ongoing challenge [150]. Within the CubeSat domain, to date only a single virtualization solution has been developed– QuickSAT/Xen [152]. However, the aforementioned solution does not delve in wireless resource virtualization. Furthermore, in the absence of published performance metrics, it is impossible to quantify the impact a Type 1 hypervisor of this kind has on system performance. A more detailed discussion about virtualization in CubeSats, and the associated challenges has been presented in Section 6.7.

6.5 Control and Management Layer

In keeping with the Management-Control Continuum (MCC) [153], IoST does not differentiate between the management and control entities. Instead, the control and management layer (CML) is responsible for network control, management, and performance optimization. The CML interfaces with the Policy Layer that guides its functioning, and the system elements that come under the purview of the CML include both the Access as well as CubeSat networks. As shown in Figure 6.6, as part of the CML, we introduce the following entities.

6.5.1 IoST Network Base

Deployed at the IoST Hub, the IoST Network Base is the network database that stores and maintains network status information. Specifically, for the CubeSat network, it stores information relating to:

- **Ground-to-satellite Links (GSLs):** IoGW-CubeSat ID pair, band of operation, link capacity, link utilization, packet error rate, MCS, SNR, transmission distance, and propagation delay.
- **Inter-Satellite Links (ISLs):** CubeSat ID pair, band of operation, link capacity, link uti-

lization, packet error rate, MCS, SNR, transmission distance and propagation delay.

- **CubeSats:** CubeSat ID, orbital plane ID, orbital altitude, computational capacity, computational load, and azimuth and elevation angles.
- **Orbital Plane:** Orbital plane ID, altitude, inclination, longitude of ascending node, and eccentricity.

For, the Access Network, it holds the following information:

- **Sensing Devices:** Location, energy level, computational capacity, and computational load.
- **Access Links:** Band of operation, MCS, packet size, packet error rate, transmit power, and link delay.

With respect to the Hub itself, the following information is stored in the network database:

- **IoST Hub:** Hub location, computational load and capacity, associated CubeSats, and associated IoGW and GSLs.
- **Network Slice:** Slice ID, tenant ID, Service Level Agreement (SLA), bands of operation, and computing resource requirements (memory, processing and storage).

6.5.2 IoST Virtualization Manager

The IVM manages the operation of the system hypervisors, and makes virtualized resources available to the Slice Controller and NOOC. The mapping of physical resources to virtual entities, and the associated lifecycle management fall under the purview of the IVM.

6.5.3 Slice Controller

IoST implements support for multi-tenancy through network slicing. More specifically, through network slicing, we seek to deploy services with differing SLA requirements over

the same physical infrastructure, in an end-to-end manner, i.e., the slice definition remains consistent from source to destination. At the infrastructure level, support for slicing comes from multi-function sensors, CubeSats with multiple sensing payloads, and the Access and CubeSat networks over which multiple isolated networks can be deployed. Within IoST, a slice is uniquely identified by its Slice ID, and is characterized by its SLA, radio resource and computing resource requirements. The slice-service relationship is modeled by a one-to-many relationship, with there being multiple possible services associated with each slice. Accordingly, resource allocation between slices is a vital network management primitive necessitating the need for a slice-neutral Slice Controller.

The Slice Controller is deployed at the IoST Hub and receives the slice definition from the Policy Layer over the NBI, which it stores in the IoST Network Base. When one or several slices are required to be deployed as indicated by the Policy Layer, the Slice Controller fetches the slice information from the network database, and then allocates infrastructure resources among the slices. In other words, the Slice Controller can be viewed as an intra-slice resource manager that guides the functioning of the IVM, instructing it to allocate non-conflicting virtual resource sets to different slices.

6.5.4 Network Orchestration and Operations Controller

While the Slice Controller handles orchestration primitives only, in keeping with MCC, the NOOC deals with both network orchestration and network control. Once the Slice Controller allocates a subset of the available resources to a slice, a NOOC instance at the IoST Hub belonging to the tenant takes over the lifecycle management of network functions within that slice, performing instantiation, scale-out/in, performance measurements, event correlation, and termination. Therefore, the NOOC works in close cooperation with the IVM and the Slice Controller, ensuring elasticity and optimal resource utilization. For example, the NOOC instance can request the Slice Controller to scale up or scale down the resources allocated to a slice based on the immediate requirements of the services utilizing that

slice. On the other hand, the NOOC also guides the placement of network functions on the underlying infrastructure through it's interactions with the IVM. Thus, intra-slice resource management is one of the key aspects of the NOOC.

In terms of network control operations, the segment of the network beyond the AcGW, through the IoGW and including the CubeSat network is under the control of the NOOC. The principal functionalities of the NOOC under the network control realm involve the following:

- **PHY Layer Functions:** Selection of frequency band, channel bandwidth, modulation and coding scheme (MCS), transmission power and number of antennas for CubeSats.
- **MAC Layer Functions:** QoS and channel aware prioritization and scheduling.
- **Network Layer Functions:** Setting the optimal packet size, and providing a list of RX candidates to choose from for transmission at each hop within the CubeSat network.

6.5.5 Access Network Controller

The Access Network Controller (ANC) is responsible for the Access Network leading up to the AcGW, i.e., it primarily exercises control over the indirect access segment sensing devices. The ANC is co-located with the AcGW, and works in close cooperation with the IoST Network Base. In a manner similar to the NOOC, we categorize the ANC's functions as follows:

- **PHY Layer Functions:** Setting the MCS and transmission power while operating under an error rate requirement constraint.
- **MAC Layer Functions:** Packet scheduling and prioritization, and optimizing sensor sleep schedules.
- **Network Layer Functions:** Packet size optimization, and routing within the sensor field.

As is apparent, a slice is provided with an ANC instance only in the event of having a indirect access segment. In the near-Earth case, the ANC's functions are taken up by the NOOC, since it controls the CubeSat network. Since each ANC is augmented with an AcGW, an ANC instance is responsible for a single Access Network. By eschewing a physical centralized control solution, and delegating control processing within the Access Network itself, we help alleviate scalability issues arising from congestion at the IoST Hub or associated IoGWs in the event of increased control traffic arising from a large number of devices. Incorporating such flexibility in our architecture allows IoST to adapt to different application scenarios while still maintaining the same generalized architectural framework.

6.5.6 Security Controller

The Security Controller is deployed at the IoST Hub, and incorporates the Security and Privacy Sublayer described next.

Security and Privacy Sublayer

The Security and Privacy Sublayer resides in the Security Controller which itself is a part of the CML. It interacts with all three layers, i.e., Infrastructure, CML, and Policy, and protects the availability, integrity, and privacy for all connected resources and information. Specifically, to protect the communications throughout the entire system and to ensure all trusted devices/data are operated/processed in a secure manner, the IoST architecture uses an identity-based networking service that utilizes flow rules to profile incoming traffic. Particularly, the IEEE 802.1X protocol is implemented for port-based network access control, and a combination of Kerberos and LDAP is utilized for authentication along with an access control server at the Security Controller that implements the Diameter protocol. The security controller also implements an Intrusion Detection System (IDS) that uses both signature-based detection via a global attack signature repository, and anomaly-based detection that detects unusual traffic behavior.

6.6 Policy Layer

While it interfaces with the IoST architecture over the NBI, the Policy Layer is not part of the core system, and is treated as an external entity. Each tenant, or service provider that makes use of IoST infrastructure has its own Policy Layer that shapes control decisions. The Policy Layer has access to the IoST Network Base, which provides it with an abstracted view of the network state. It issues policy directives to the CML over the NBI, and the CML incorporates the policy while formulating network control strategies.

An example of a policy directive could be avoiding a specific route in the CubeSat network because of higher monetary costs associated with it, or restrictions on the nodes over which the NOOC can instantiate network functions, in terms of location and resource utilization levels. We do not intend to provide Policy Layer specifications, instead relying on the adherence of the layer implementation to a standardized NBI. In this manner, IoST can handle a variety of tenant requests in a service-oriented manner. Furthermore, access to the IoST Network Base over the NBI allows service providers to customize data acquisition that can be then used for analysis of network trends that are fed back to the policy framework.

6.7 System Procedures

In this section, we discuss the key system procedures that lend IoST its novelty, and the primary challenges associated with each. At the outset, we note that unlike terrestrial wireless networks, the space environment is characterized by long propagation delays and an ever-changing network topology. Consequently, as explained in the following sections, a variety of terrestrial SDN concepts are rendered sub-optimal in their applicability to IoST, and there is a need for novel solutions to tackle the unique challenges faced by the system.

6.7.1 Joint Optimal Physical-Link Layer Resource Allocation with vCSI

The need for resource allocation techniques that jointly optimize across the physical and link layers is born out of the large number of parameters that characterize a transmission link in IoST– frequency bands, number of antenna elements, MCS options, and transmission power and bandwidth. Furthermore, the global presence of IoST gives rise to a large number of simultaneous flows, necessitating the need for a link scheduling algorithm that can achieve throughput optimality.

In addition to the computational complexity associated with cross-layer optimization, within the context of IoST, resource allocation faces two major challenges: (i) the aforementioned complications arising from the space environment, and (ii) absence of control capabilities on the CubeSats. Furthermore, unlike terrestrial networks, where there is only a single layer of wireless access nodes (for example, Remote Radio Heads or Distributed Units), in IoST the entire network is wireless, and therefore resource allocation must be done at each hop, adding to the problem. Since the NOOC on the ground must perform resource allocation for the CubeSats in the space, the absence of real-time Channel State Information (CSI) at the NOOC poses a major challenge. Owing to limited on-board processing on CubeSats, we have already ruled out partial delegation of control functionalities.

To this end, we introduce joint physical-link layer resource allocation augmented with virtual CSI (vCSI) in IoST. In a vCSI-based approach, the NOOC runs a simulation of the IoST system which in turn makes use of prediction algorithms to generate vCSI. Then, the NOOC makes use of the generated vCSI as the basis for its decisions. Furthermore, the vCSI prediction is periodically augmented with up-to-date CSI to enhance prediction accuracy. More specifically, the practical realization of this system requires the development of an online CSI prediction scheme that takes into account metrics including but not limited to time, frequency band of operation, CubeSat position, and noise temperature.

6.7.2 Tackling Long delays and Temporal Variation in Network Topology Through Stateful Segment Routing

IoST is a classic example of a Long Fat Network (LFN), characterized by a large bandwidth-delay product which renders the traditional SDN approach sub-optimal. First, we note that the data plane in traditional SDN systems is stateless, i.e., the forwarding function of the switch is based on the match-action model which makes use of various packet header fields, such as the Ethernet source address or IPv4 destination address to match incoming flows to the corresponding entries in the flow table of the switch. The flow table entries in turn are set by the controller, which in this case happens to be the on-Earth NOOC. Consequently, the temporal variation in network topology causes flow table updates to become obsolete by the time they reach those CubeSats that are farther away from the NOOC. Second, we note that terrestrial reactive forwarding [154] in use across a majority of the SDN systems is ill-suited for IoST owing to the large volume as well as frequency of control traffic. Traditionally, techniques such as control traffic balancing [109] have been used to great effect for efficient management of control traffic, however, they are not inherently applicable to LFNs because of the prevailing delays in the environment.

To this end, we introduce the concept of Stateful Segment Routing (SSR). SSR has been envisioned with a view to overcoming the drawbacks of the stateless data plane, and minimizing control traffic. Segment Routing (SR) functions by dividing the path to be traversed into a sequence of logical segments, with a set of middlepoints interconnecting successive segments [155]. In adapting SR for use in IoST, we note that it leads to a minimization in control traffic as the flow table entries are greatly reduced, as a result of the reduced set of forwarding entries that can be used for all flows that share a common middlepoint. Furthermore, while SR constructs the logical paths (segments) between middlepoints, we recognize that due to the ever-changing network topology, the next hop at each CubeSat, as decided by the NOOC may not always be reachable. Therefore, in IoST we propose the use of a stateful data plane, where forwarding is done based not only on the matching of

header fields, but also on the system state, where state is defined a function of the network topology with time. The deterministic nature of the topological variations allows for accurate state characterizations, and the NOOC pre-emptively determines the best route for a number of such states, and then each CubeSat can select the appropriate next hop, based on a combination of match fields and state

A major challenge in implementing SSR lies in ensuring that flows are routed through paths that have a large number of middlepoints in common. Optimal middlepoint selection and path computation remain challenging even in the case of terrestrial wired networks. Additional complications introduced by a stateful data plane further make the problem even more difficult. Despite this, the proposed heuristics must be efficient and scalable in order to maintain a low control plane response time.

6.7.3 Robust Connectivity Through Optimal IoST Hub Geolocations

Given the small footprint of a single CubeSat, the need for continuous coverage necessitates the deployment of a CubeSat constellation, with multiple orbital planes, and multiple CubeSats within each plane. However, robust forwarding of control policies requires the presence of a low-latency control path. While easy to achieve in terrestrial wired and wireless scenarios, due to the inherent nature of the medium hop-by-hop forwarding of control messages in a CubeSat constellation is affected by high link latency. On the one hand, attempting to maintain a Line of Sight (LoS) link with each CubeSat in the constellation will make the number of IoST Hubs prohibitively large both from a cost as well as network management perspective. On the other hand, a single IoST Hub would result in extremely high convergence times affecting both network throughput and latency. A suitable middle ground can be found by having a limited number of IoST Hubs, and an additional number of IoGWs in order to implement gateway diversity.

To address this issue, we propose the IoST Hub placement problem with multiple geodistributed locations. The aim is to minimize both the number of hubs required, and the

control traffic convergence time across a large network. More specifically, with regard to the temporal variation in LoS CubeSats, the controller placement problem should determine: (i) the number of required IoST Hubs and their individual geo-locations, (iii) the number of IoGWs under each hub and their geolocations, and (iii) the control domain assignments for each hub. Solving an optimization problem of this kind is very challenging because it is NP-hard along with tremendous variables. It is impossible to solve and obtain the optimal values in a time-efficient manner (i.e., even finding a feasible solution will require a certain amount of computational time). To counter this challenge, it is imperative to develop a fast approximation algorithm. Techniques such as LP relaxation, pre-processing, scaling, and randomized rounding could prove useful in this context.

6.7.4 Synchronization of Geo-Distributed IoST Hubs

Inter-controller communication and synchronization across hubs poses another major challenge. In order to achieve logical centralization in the aforementioned distributed environment, it is necessary to ensure that all IoST Hubs maintain the same global network view. The problem becomes even more challenging in long-delay environments that change with time, i.e., the network state changes at a rate faster than the time taken by the distributed controllers to converge to the same state, as a result only partial synchronization may be achieved. Further complications arise due to the fact that control domain assignments are a function of time as well traffic, unlike the wired case.

Additionally, a major task is measuring the effectiveness of the synchronization solution, and its dependence on the network topology, which in the case of IoST spans both terrestrial and near-Earth domains, and examining the reliability of any such proposed solution. Furthermore, the synchronization solution should be resilient to hub failures.

6.7.5 Proactive Handovers Through GSL Outage Forecasting

The GSLs between the IoST Hub and CubeSats are vital to connectivity as they form the first hop for both control and data traffic. GSL outages can be broadly classified into two categories: (i) outages due to CubeSat mobility, and (ii) outages due to atmospheric effects such as molecular absorption, rain attenuation, cloud attenuation, and scintillations. An outage event is characterized by a dip in the SNR below a predefined threshold value that causes an interruption in data transfer. Consequently, we wish to preempt these link interruptions to maintain a high level of system reliability. At the outset, it is relatively easy to predict outages due to mobility because the movement of CubeSats is deterministic in nature. On other hand, outage events due to atmospheric effects are stochastic in nature, and consequently more difficult to predict. Thus, the development of low-complexity link outage prediction algorithms that run over the NOOC represents a major challenge in this context.

However, link outage prediction only solves half the problem, with the handover being the other half. Clearly, if the IoST Hub is experiencing an outage at a given time, any attempt to establish GSLs with CubeSats in its domain will fail. However, IoST employs gateway diversity to ensure that the atmospheric effects across all IoGWs are uncorrelated. Furthermore, satellite diversity is enforced through the fact that each IoGW is capable of establishing connectivity to different sets of CubeSats which are not necessarily overlapping. Therefore, the most suitable candidate IoGW is characterized not only by high link SNR, but also by the CubeSats it can communicate with in terms of the proximity of said CubeSats from the destination. The principal idea here is to offset the handover interruption time by appropriate selection of the candidate IoGW. Once the candidate IoGW has been selected, the IoST Hub forwards data to it, which is then delivered to the CubeSat network.

6.7.6 Lightweight Hardware Virtualization for CubeSats

At the outset, CubeSats require virtualization of computing, storage, memory, and radio resources. However, the hypervisors that are commonly used in terrestrial networks have been designed for server-grade hardware. Consequently, the virtualization overhead introduced by established hypervisors makes them unsuitable for use in resource constrained devices such as CubeSats [48]. To this end, IoST must also develop lightweight virtualization solutions for CubeSats, that support radio resource virtualization with minimal overhead. More specifically, a large number of CubeSats today make use of ARM-based microprocessors [156]. With both Docker and LXC adding ARM support in recent years, there is a strong case for containerization in CubeSats. However, many aspects of container networking are not well understood [157]. Therefore, before full-fledged containerization solutions are developed for CubeSats, the initial challenge is to quantify the impact of IoST traffic on container networks.

6.7.7 Automated Device Provisioning Through ANC

The IoST architecture provides the ANC for sensor device control, which we leverage for plug-and-play operation. Manual configuration and re-configuration of sensing devices is a time consuming affair, and a significant impediment to network scalability, moreso in the case of virtual sensor networks. Therefore, the ANC must provide a mechanism to enable zero-touch provisioning. The use of SDN is particularly beneficial to this end, as it allows for dynamic network reconfiguration through custom control functions such as task scheduling and energy management in resource-constrained sensing devices.

Task scheduling has become increasingly important due to the proliferation of multi-function sensing devices. Each task is associated with a different application having its own sensing frequency, accuracy, and resolution. As devices join and leave the network, a major challenge is determining which tasks shall be assigned to each of them, and in what order, constrained by the sensing capability and resource availability of the sensing device,

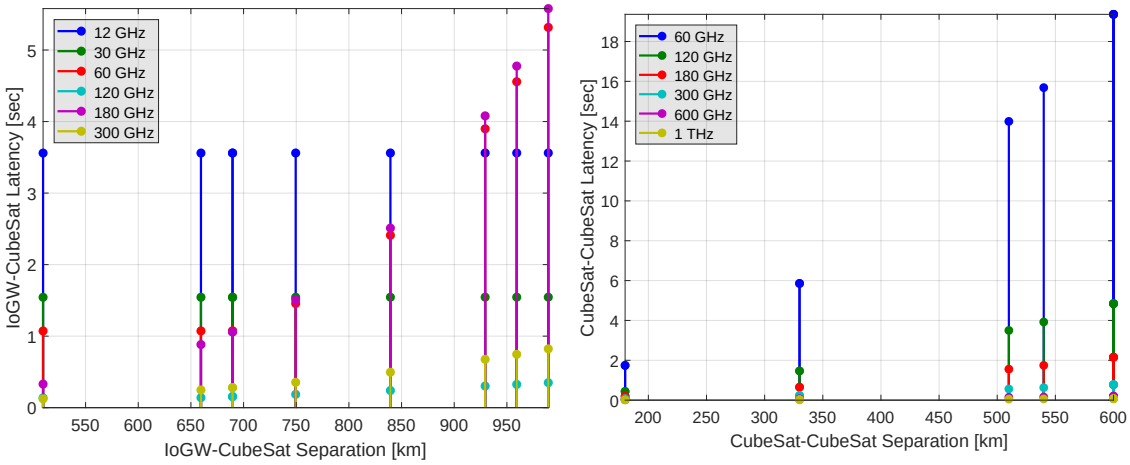
Table 6.1: Simulation parameters for evaluating IoST system performance

Parameter	Value
Constellation Configuration (Altitude [km], No. of CubeSats per Plane, No. of Planes)	$(500, 71, 36)$, $(600, 72, 37)$, $(700, 73, 37)$, $(800, 74, 38)$, $(900, 75, 38)$
Carrier Frequencies	Low-band (3, 12 GHz), mmWave (30, 60 GHz), THz (120, 180, 300, 600, 1000 GHz)
Transmit Power (at CubeSat and IoGW)	10 W
Average Noise Temperature (IoGW-to-CubeSat, Exosphere)	300 K, 1500 K
CubeSat Antenna (Type, Diameter, Efficiency)	Parabolic reflector, 10 cm, 50%
Lower Bound on SNR	10 dB
OpenDaylight Operation	In-band Control
Channel Bandwidth Threshold (Carrier, Threshold)	(3, 0.06 GHz), (12, 0.65 GHz), (30, 1.5 GHz), (60, 2.16 GHz), (120, 17 GHz), (180, 30 GHz), (300, 50 GHz), (600, 100 GHz), (1000, 150 GHz)

and the requirements of each task. Note that the heterogeneity of the networks and various QoS requirements make the scheduling and coordination of endpoint resources in IoST complex. Additionally, for the indirect access segment involving the IoGW, pre-processing and analysis could be performed at the IoST Hub, if necessary, to minimize bandwidth consumption in the CubeSat network.

6.8 System Performance Evaluation

In this section we establish the performance baseline for the IoST system along three primary domains: (i) single-hop link metrics, (ii) next-hop link metrics, and (iii) overall system performance. A more detailed version of the system performance results which have been summarized herein can be found in [6]. In particular, we focus on metrics including but not limited to link throughputs, link latencies, handover durations, and next-hop candidates. The IoST system is implemented using the Systems Toolkit (STK), the Open vSwitch (OVS) virtual switch, and the OpenDaylight (ODL) controller operating in the in-band mode. Table 6.1 represents the system parameters used in the evaluation.



(a) GSL latency.

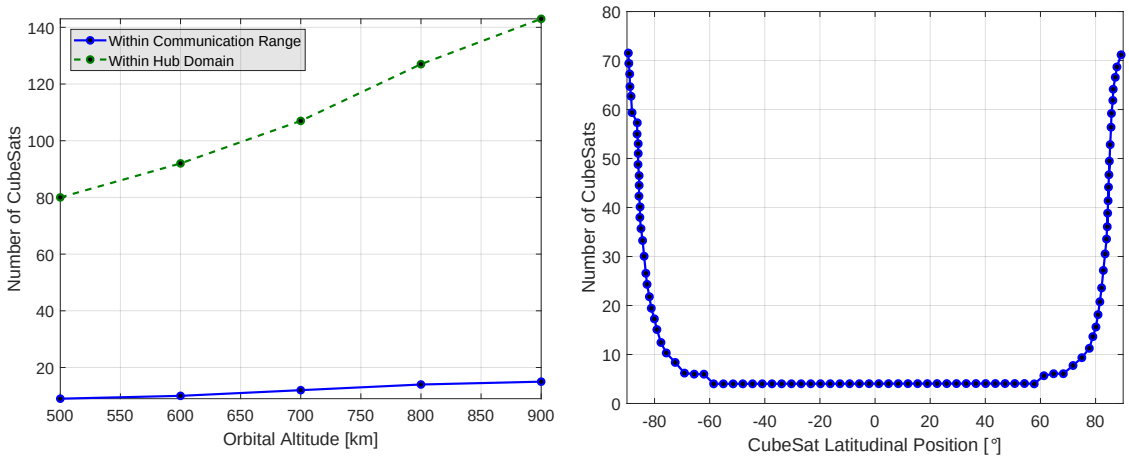
(b) ISL latency.

Figure 6.8: Single-hop metrics.

6.8.1 Single-Hop Link Metrics

Within the single-hop metrics, we direct our attention to the GSL and ISL link latencies as shown in Figures 6.8a and 6.8b. Beginning with Figure 6.8a, we note that it represents the variation in delay of the link between IoGW and a CubeSat with increasing distance. The scenario under consideration assumes a uniform water vapor density of $7.5\ \mathrm{g/m^3}$, and constellation orbital altitude of $500\ \mathrm{km}$, with a gigabyte of data to be transferred from the IoST Hub to the CubeSat. On an average, there are 9 GSLs active at a given time, each being represented by a stem in the plot. Then, for each of the 9 links, we vary the carrier frequency from 3 to $300\ \mathrm{GHz}$, and observe the change in latency. At the outset, we do not show the $3\ \mathrm{GHz}$ carrier due to its low data rate of $20\ \mathrm{Mbps}$ even at a relatively low separation of $500\ \mathrm{km}$.

From Figure 6.8a, we identify two general trends– first, the link latency decreases with increasing carrier frequencies, and second, the latency increases with increasing distance. The first result follows from the fact that larger carrier frequencies are able to support higher data rates, which in turn causes the transmission delay to fall, while the second result is attributed to falling data rates as a consequence of decreasing SNR, and increasing propagation delays with increasing distance. For example, for a GSL separation of $510\ \mathrm{km}$,



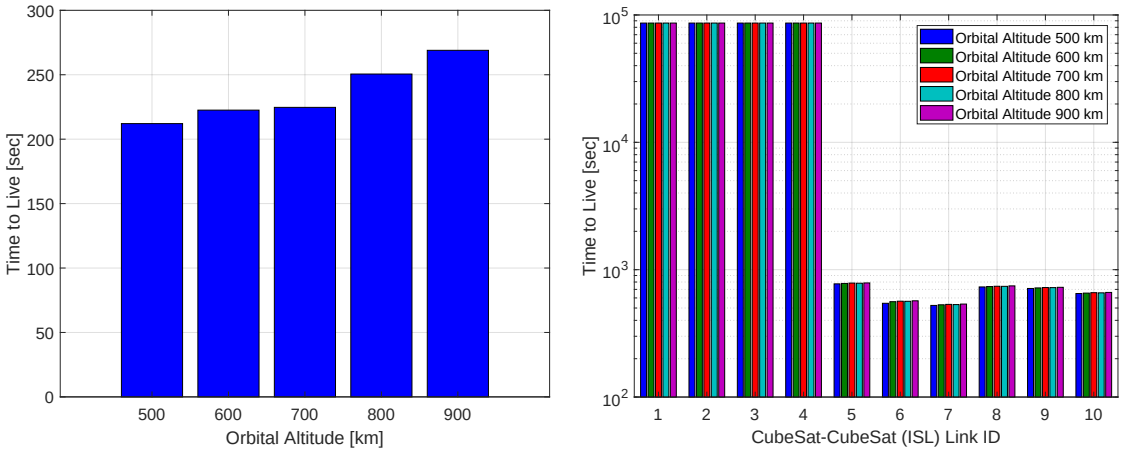
(a) Number of CubeSats relative to IoST Hub.

(b) Available next-hop CubeSats.

Figure 6.9: Next-hop metrics: number of CubeSats.

the link latency falls from 3.52 s at 12 GHz to 0.10 s at 300 GHz, while for a 990 km GSL the latency rises to 0.91 s for the same 300 GHz carrier. In addition, we also note an anomaly that becomes apparent at distances close to 900 km. The 60 and 180 GHz exhibit the worst performance as a result of molecular absorption due to oxygen and water vapor respectively.

Figure 6.8b represents the change in latency for an ISL with change in the CubeSat separation. Since the water vapor density in the exosphere is negligible, we do not consider its impact on the link delay. Furthermore, in the absence of molecular absorption, the results obtained follow the general trend, and do not show any anomalies. A link separation of 330 km results in delays of 0.05 s and 5.85 s at 60 GHz and 300 GHz respectively, increasing to 19.36 s and 0.07 s for a separation of 600 km. In other words, an increase in the data rate due to an increase in the carrier frequency causes a decrease in the average link latency, and increasing link separations result in rising link latencies due to increasing propagation delays, and falling data rates. Furthermore, we note that there an average 10 active links regardless of the constellation configuration.



(a) Link duration relative to IoST Hub.

(b) Link duration for ISLs.

Figure 6.10: Next-hop metrics: link duration.

6.8.2 Next-Hop Link Metrics

The next-hop link metrics include parameters such as the number of available next hops, the number of CubeSats within a Hub's control domain, and the duration for which a link is active before a handover takes place. These metrics help quantify the network toplogy, and play a vital role in system modeling. Figure 6.9a represents the variation in the total number of CubeSats within a IoST Hub's domain with change in orbital altitude. The figure also shows the number of CubeSats that satisfy the minimum acceptable SNR constraint, and with which the IoST Hub can exchange control and data traffic. Both metrics are characterized by an increasing trend with increase in orbital altitude. For example, at an altitude of 500 km, there are 80 CubeSats within the IoST Hub's domain, 9 of which meet the minimum SNR requirement. As the altitude increases to 900 km, the total number of CubeSats increases to 143, 15 of which meet the SNR constraint. This increase can be attributed to the greater number of CubeSats in higher altitude constellations.

From Figure 6.9b, we note the number of available next hops for a given CubeSat, and its variation as the CubeSat revolves around the Earth. At the poles, where the CubeSat density is the highest, the number of available candidate next hops exceeds 70, a figure which falls steeply as the CubeSat moves towards the equator where the satellites are the

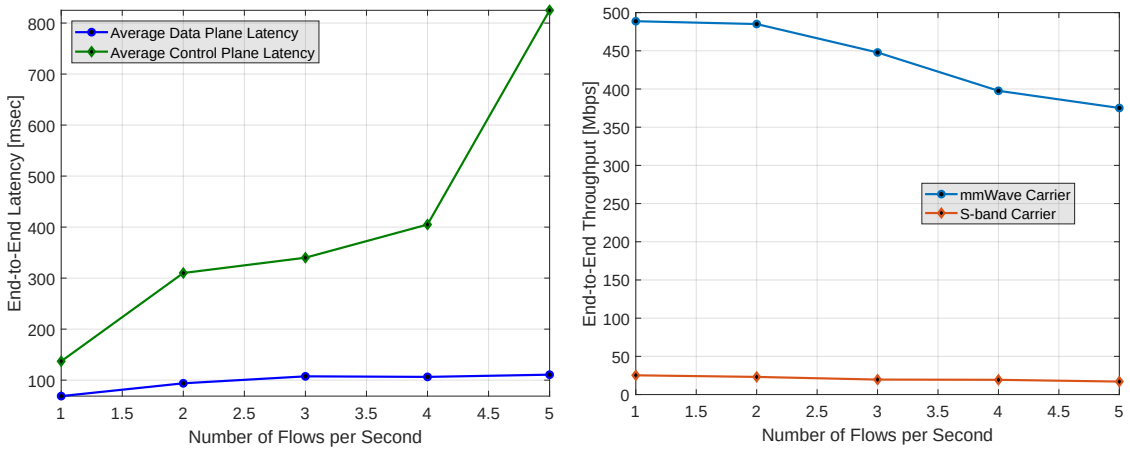
farthest apart. For, latitudes between -60° and $+60^{\circ}$, the number stays constant at 4 which can be attributed to the fact that only the adjacent CubeSats are within $610\ \mathrm{km}$ of each other, which is the distance at which SNR $= 10\ \mathrm{dB}$.

Figure 6.10a represents the average link duration before a handover takes place. More specifically, Figure 6.10a shows the variation in the average duration for which a GSL is active as a function of the orbital altitude. The general trend observed in Figure 6.10a is that the duration increases with an increase in orbital altitude. This result is the outcome of the fact that CubeSats in higher altitudes move more slowly due to a lower orbital velocity. For example, the duration increases from $210\ \mathrm{s}$ for an orbital altitude of $500\ \mathrm{km}$ to $270\ \mathrm{s}$ at $900\ \mathrm{km}$. In a similar vein, Figure 6.10b shows the average duration of an ISL. From the figure we note that 4 of the 10 links are always active, and from our understanding of Figure 6.10a we realize that these links correspond to the adjacent CubeSats. For the remaining 6 links, the access duration is in the range of 550 – $750\ \mathrm{s}$.

6.8.3 Overall System Performance

In order to evaluate the overall system performance, we set up a test scenario that involves data delivery between Atlanta and Lisbon over an IoST constellation deployed at an orbital altitude of $500\ \mathrm{km}$. A single IoGW is co-located with the IoST Hub deployed in Atlanta. The GSL operates at $60\ \mathrm{GHz}$ in moist air, and the ISL also operates at $60\ \mathrm{GHz}$. On the other hand, OpenDaylight operating in reactive mode with in-band control serves as the network controller in the test setup. In addition, we have made use of a static topology, which we consider as a fair assumption in light of adjacent CubeSats always being in contact with each other. Next, in addition to the Atlanta-Lisbon (A-L) flow, we generate a number of flows, ranging from 1 to 5 per second, with a random source-destination pairing within the topology, and measure the latency and throughput performance of the A-L flow under these conditions.

The latency performance is characterized by two metrics– data plane latency and control



(a) End-to-end link latency.

(b) End-to-end link throughput.

Figure 6.11: End-to-end system operation.

plane latency. In particular, control plane latency refers to the time between the transmission of a "PacketIn" message by the CubeSat, and the reception of the "FlowMod" message sent by the IoST Hub, as shown in Figure 6.11a. As the number of flows per second increase, both control and data plane latency values show an increase, reflecting the higher amount of traffic prevalent in the network. In particular, the control plane latency experiences a sharp increase due to there being a single IoST Hub in operation whose GSL becomes increasingly congested as the number of simultaneous flows in the network rise, each of which direct control traffic over the same link. Figure 6.11b compares the end-to-end throughput performance for the mmWave and S-band carriers. As the number of flows in the network rise, the throughput suffers as the same bandwidth is divided among an increasingly large number of flows. Furthermore, by virtue of its larger channel bandwidth, the mmWave carrier performs nearly 15–20 times better than the S-band carrier, reaching a peak throughput of nearly 500 Mbps.

6.9 Highlights

This chapter of the thesis presents a ubiquitous cyber-physical system, the Internet of Space Things, a paradigm-shift network architecture that will significantly change the way IoT

and cellular networks are expected to operate. IoST expands the functionalities of the traditional IoT, by not only providing an always-available satellite backhaul network, but also by contributing real-time satellite-captured information and, more importantly, performing integration of on-ground data and satellite information to enable new applications. The fundamental building block for IoST is a new generation of CubeSats, which are augmented with SDN and NFV solutions. The use of SDN and NFV allows us to introduce novel and innovative concepts such as vCSI, SSR, and satellite diversity that are purpose-built for tackling the long delays and topological variations that characterize the space environment. Additionally, the major open problems which are critical to a full deployment of CubeSat-based IoST are identified. A system performance evaluation covering single-hop, next-hop, and end-to-end metrics further cements the potential of IoST. In this manner, IoST helps realize pervasive end-to-end global connectivity.

CHAPTER 7
LARGE-SCALE CONSTELLATION DESIGN FOR ULTRA-DENSE CUBESAT NETWORKS

In Chapter 6 we introduced a new cyber-physical system centered around CubeSats, known as the Internet of Space Things. The first challenge in realizing a system of this kind relates to the development of a robust network topology that can meet the coverage and connectivity requirements associated with a wide range of use cases. With a view to achieving this goal, this chapter of the thesis focuses on the aforementioned topology design problem.

7.1 Motivation

Satellite communications has been recognized as a key component of 5G systems for establishing remote connectivity [56], and is widely expected to play an increasingly ubiquitous role in the next-generation 6G wireless systems [1]. The proliferation of satellite communications solutions has been driven in large part by the availability of low-cost launch opportunities, and advances in satellite hardware design allowing for the use of commercial off-the-shelf (COTS) components. Together, these two factors have led to the democratization of space, with startups, research institutions, universities, and even high schools emerging as important stakeholders. In particular, CubeSats are being seen as a promising solution for realizing robust connectivity at low costs.

To this end, we have previously introduced the concept of the Internet of Space Things (IoST), a cyber-physical system spanning air, ground, and space, in Chapter 6. A ubiquitous system of this kind requires optimized coverage and consistent connectivity. For example, certain use cases may necessitate global coverage, while others may require targeted region-specific coverage. Thus, optimal constellation design is of great importance to mission planners, the significance of which comes from the fact that it has a direct impact on the

system's cost, scalability, and efficacy.

Conventional low Earth orbit (LEO) constellations are typically characterized by the presence of fewer than a hundred satellites, and as such cannot meet the needs of systems such as IoST. Motivated by the need for improved coverage, reliable connectivity, and increased redundancy, mega-constellations of several hundred satellites have gained significant traction over the past two years [158]. In general, constellation design typically involves solving for several inter-related parameters such as: (i) the apogee and perigee radii, (ii) the orbital eccentricity, (iii) the number of CubeSats per orbital plane, (iv) the number of orbital planes, and (v) the initial longitude of the ascending node, argument of perigee, and true anomaly of the constituent CubeSats. While a fairly challenging problem in itself, the presence of an extremely large number of satellites further serves to complicate the problem. As described in Section 7.2.1, the existing state-of-the-art constellation design frameworks are largely geared towards the design of systems with a few dozen satellites at best. Consequently, there is a pressing need for a large-scale constellation design framework that can scale well for several hundred CubeSats.

At the outset, we intend to address a two-fold objective through this chapter. First, we will design a highly customizable large-scale constellation design optimization framework for CubeSats, and then, we will use the proposed framework to design a set of constellations for IoST. With a view to address the plethora of use cases that IoST is expected to serve, the framework presented herein can be used for the design of constellations geared towards either global coverage, latitude-specific coverage, or regional coverage. Furthermore, it also provides the constellation designer with the freedom to set the desired level of connectivity among the constituent CubeSats, as required.

As part of the design framework, we first implement a fast and accurate orbit propagator that accounts for perturbations arising from the Earth's oblateness through the J_2 spherical harmonic coefficient. Additionally, we verify the accuracy and computing efficiency of the propagator through extensive comparisons with the state-of-the-art Systems Tool

Kit (STK) [159] software. Next, we quantify the coverage achieved by the constellation through metrics that involve the use of geodetic positions of the sub-satellite points (SSPs), along with spherical Voronoi diagrams and Delaunay triangulations. We also take into consideration the level of connectivity between CubeSats that form the constellation. Robust connectivity characterization is achieved by using the number of active inter-satellite links (ISLs) as the connectivity metric. Furthermore, we note that since IoST is limited to the exobase [7], with a target deployment altitude that ranges from $600\ \mathrm{km}$ to $1000\ \mathrm{km}$ above the Earth's surface, the maximum possible orbital eccentricity that can be achieved is 0.02, which corresponds to an apogee and perigee radii of $7378\ \mathrm{km}$ and $6978\ \mathrm{km}$ respectively. Since even the largest possible eccentricity leads to a nearly circular orbit, the optimization framework presented herein is intended for the design of circular orbits in particular. Furthermore, we focus exclusively on designing uniform Walker constellations owing to their vast popularity in the domain of satellite-based IoT and broadband services.

More specifically, we wish to determine: (i) the orbital altitude, (ii) the orbital inclination, (iii) the number of CubeSats per orbital plane, (iv) the number of orbital planes, and (v) the relative phasing between CubeSats in different planes, while minimizing the number of CubeSats in the constellation, and maximizing the coverage and connectivity related metrics. A combinatorial optimization problem of this kind does not admit an easy exact solution, and, therefore, we make use of the simulated annealing (SA) meta-heuristic to obtain an approximate solution. In doing so, we note that conventional annealing functions are not applicable to the proposed constellation design problem because of additional hidden constraints on system parameters. To this end, we have developed a custom annealing function which ensures that every generated solution set is feasible.

Furthermore, we also apply the proposed framework to design a set of constellations for IoST targeting both global as well as latitude-specific and regional use cases. More specifically, for the global coverage use case, we compare the obtained IoST constellation with existing state-of-the-art satellite-based Internet of Things (IoT) services. On the other

hand, for the latitude-specific and region-specific cases, we examine the impact of changing system parameters on the constellation design. To summarize, the major contributions of this chapter are as follows:

- Novel constellation coverage characterization based on the geodetic positions of the SSPs, along with spherical Voronoi tessellations, and Delaunay triangulations, followed by CubeSat connectivity characterization based on ISL availability.
- A scalable combinatorial optimization framework based on simulated annealing (SA) that provides the optimal orbital altitude, orbital inclination, number of CubeSats per orbit, and number of orbital planes in the constellation, along with the relative phase difference between CubeSats in adjacent planes.
- A set of constellation designs, each catering to a specific use case, i.e., global, latitude-specific, or region-specific, along with a detailed set of results that includes a performance comparison with existing global connectivity solutions, and examines the impact of system parameter variation for the presented coverage cases.

To the best of our knowledge, this work is the first to present a highly-customizable large-scale constellation design optimization framework that takes into consideration both coverage and connectivity parameters, and adapts to different operational scenarios. The remainder of this chapter is organized as follows. After discussing the related work in Section 7.2, in Section 7.3, we present the different subsystems of the framework, namely, the orbit propagation module, and the coverage and connectivity estimation modules. Then, in Section 7.4, we present the SA-based optimization framework, followed by a variety of constellation designs for IoST, obtained using the proposed framework, in Section 7.5. Finally, we conclude the chapter in Section 7.6.

7.2 Related Work

Satellite-based IoT networks have gained significant traction in the past couple of years with several commercial solution providers in various stages of deployment. Our interest in this segment of the market is primarily limited to services that involve the use of CubeSats for reasons outlined in Section 7.1. Accordingly, the prior art in this domain can be broadly divided into two categories– state-of-the-art constellation design frameworks and state-of-the-art CubeSat constellations. Both categories are highly inter-related as CubeSat-based IoT services have much to benefit from optimized constellation design.

7.2.1 State-of-the-Art Design Tools and Frameworks

At the outset, the estimation of coverage and connectivity metrics is vital to the constellation design problem. Accordingly, there exist a number of tools that are geared towards constellation coverage and connectivity analysis, with ASTROLIB [160], REVISIT/COVERIT [161], and STK [159] representing the commonly used options. ASTROLIB and REVISIT are both proprietary tools developed by The Aerospace Corporation, and are not available to the general public. The former is a software library supporting orbital mechanics, satellite visibility, and general-purpose mathematical functions, while the latter is designed to compute performance metrics such as revisit time, response time, access interval duration, daily visibility time, and daily number of accesses. On the other hand, STK from Analytical Graphics Inc., is available as a COTS software tool to the general public and includes several modules for Earth coverage and constellation connectivity calculation. However, it does not integrate well with constellation design optimization frameworks owing to the computation time associated with analysis of large-scale constellations [162, 163]. We also note the presence of open-source tools such as the General Mission Analysis Tool (GMAT) [164], poliastro [165], and the JuliaSpace Satellite Toolbox [166], however, these solutions lack support for metrics such as revisit time and ISL availability.

Furthermore, we note that a majority of the prior art in the domain of constellation design frameworks has been limited to a few dozen satellites at best [167, 168, 169, 170, 171, 172, 173, 174, 162]. Additionally, these frameworks are largely centered around the commercial software packages described previously, along with the use of genetic algorithms. We have previously noted the suboptimality of existing commercial tools when it comes to the constellation design problem. More specifically, [167] leverages the coverage module of STK as the basis for the presented constellation optimization framework. While the presented work targets regional coverage, it takes into consideration a maximum of only 16 CubeSats spread across two planes. Furthermore, Marinan et al. [168] propose a staggered approach to CubeSat constellation design, wherein the constellation is built up over time using available rideshare launch opportunities. Here too, the coverage evaluation is performed using STK, with the authors presenting the optimal constellation configuration for best coverage. In addition to the absence of ISLs, we note that the analysis in [168] is limited to 36 CubeSats only. On the other hand, the Trade-Space Analysis Tool for Constellations (TAT-C) [162] introduces a new coverage calculation module that does not rely on STK, however, the constellation designs presented in [162] do not exceed a maximum of 12 satellites.

In addition, [169, 170, 171, 172] make extensive use of genetic algorithms. On the one hand, [169, 171, 172] are all geared towards global coverage, on the other hand, [170] optimizes for regional coverage. While the prior art surveyed thus far has focused primarily on coverage, Liu et al. [171] also include ISLs in their system model, albeit limited to satellites in adjacent planes only. Here too we note that the number of satellites under consideration does not exceed 60 in the best case. However, such a small number of satellites is not sufficient for optimized coverage and connectivity within the context of large-scale systems. Finally, catering exclusively to CubeSats, [173] presents a number of constellation designs for global coverage at varying orbital altitudes. While the analytical solutions provided in [173] do not suffer from issues relating to scalability, in the absence of an optimization

Table 7.1: Constellation configurations for CubeSat-based IoT and broadband services

Constellation	**Astrocast [31]**	**Fleet [32]**	**Kepler [33]**	**Aistech [34]**	**Myriota [35]**	**Planet Dove [36]**	**Lacuna Space [37]**
Purpose	IoT and M2M	IoT	Satellite backhauling	IoT, M2M, asset tracking	IoT	Earth imaging	IoT and M2M
ISL Capability	Yes	No	Yes	NA	NA	No	No
Orbital Altitude	575 km	580 km	600 km	500 km	800 km	420 km	500 km
Orbital Inclination	97.4°	NA	98.6°	97.8°	NA	52°	NA
Number of Satellites	64	100	140	102	50	150	32
Number of Orbital Planes	8	20	7	NA	NA	NA	11
Form Factor	3U CubeSat	12U CubeSat	3U CubeSat	2U CubeSat	3U CubeSat	3U CubeSat	6U CubeSat

framework it is difficult to characterize the efficacy of the presented work.

To summarize, we note of the following shortcomings in the existing state-of-the-art design solutions: (i) reliance on computation heavy commercial tools, (ii) lack of consideration for connectivity within the constellation, and (iii) absence of solutions for large-scale constellation design. To this end, the design framework proposed herein aims to overcome these shortcomings. We note that our aim is not to provide a replacement for STK, but, instead develop a rapid prototyping tool for large-scale constellation design.

7.2.2 State-of-the-Art CubeSat Constellations

In this section we survey the current satellite-based broadband and IoT connectivity landscape with an emphasis on services that use CubeSats. The past year has witnessed the launch of several communications and IoT focused CubeSats, chief among them have been satellites from Astrocast [31], Fleet Space Technologies [32], Kepler Communications [33], Aistech [34], Myriota [35], Planet Labs [36], and Lacuna Space [37]. Details concerning the constellation design of each service have been listed in Table 7.1. Furthermore, we note that with the exception of Planet, the solutions in Table 7.1 are not commercially operational yet.

In addition, we also take into consideration the Iridium NEXT constellation that consists of 66 LEO satellites deployed at an altitude of $780\ \mathrm{km}$ in 6 orbital planes at an inclination of 86.4° [175]. At the same time, we note that the aforementioned constellations are characterized by a maximum of $100 - 150$ satellites, and therefore do not fall into the class of mega-constellations which contain several hundred satellites. Instead, examples of mega-constellations include the Starlink constellation from SpaceX [39], the OneWeb constellation [176], and Project Kuiper from Amazon [177]. More specifically, the first phase of the Starlink constellation is characterized by 24 orbital planes, each containing 66 satellites, at an altitude and inclination of $550\ \mathrm{km}$ and 53° respectively, while the OneWeb constellation contains a total of 648 satellites across 18 orbital planes at an altitude of $1200\ \mathrm{km}$ and an inclination of 87.4°. However, these constellations do not contain CubeSats, and are therefore not listed in Table 7.1. Furthermore, we note that OneWeb does not make use of ISLs, which may prove detrimental to connectivity [176]. On the other hand, since Project Kuiper is yet to receive approval from the Federal Communications Commission (FCC), their constellation configuration is subject to change, and will therefore not feature in the comparison detailed in Section 7.5.

7.3 System Modules

The constellation design framework presented in this chapter follows a modular approach as shown in Figure 7.1. As shown in the figure, first, we present the orbit propagation subsystem in Section 7.3.1, which uses the static parameters as input. Then, we present the coverage and connectivity subsystems in Sections 7.3.2 and 7.3.3 respectively, with both modules using the output of the orbit propagation subsystem as input. Finally, we present the SA-based design optimization framework in Section 7.4, which uses the coverage and connectivity estimation results as input, and provides the optimal constellation configuration as output. Furthermore, as noted in Section 7.1, since IoST operates in the exosphere, the maximum possible apogee radius is $7378\ \mathrm{km}$, and minimum feasible perigee radius is

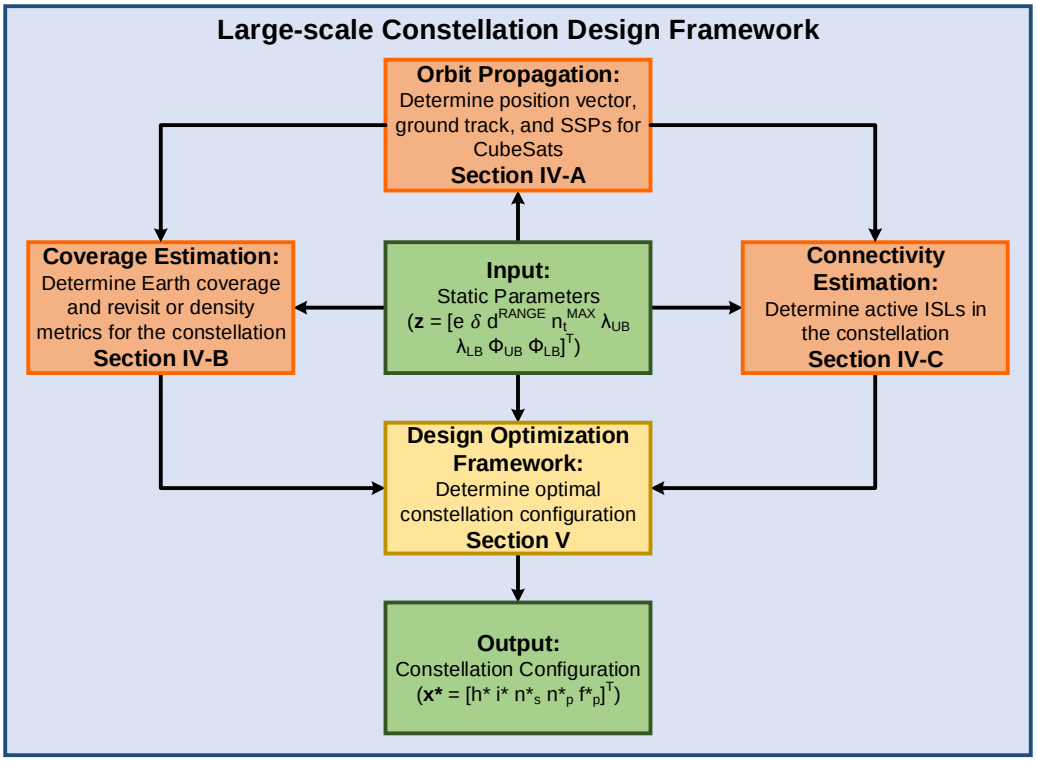


Figure 7.1: The proposed constellation design optimization framework[1].

6978 km, leading to a maximum eccentricity of 0.02. Therefore, we specifically cater to the design of uniform constellations with circular orbits, i.e., Walker constellations, as depicted in Figure 7.2. In the interest of tractability, we define three classes of system parameters– design variables ($\boldsymbol{x}$), internal variables ($\boldsymbol{y}$), and static parameters ($\boldsymbol{z}$).

The design variables refer to the constellation parameters which we wish to optimize, i.e., $\boldsymbol{x} = [h\ i\ n_s\ n_p\ f_p]^T$, representing the orbital altitude, the orbital inclination, the number of CubeSats per orbit, the number of orbital planes, and the phasing parameter of the constellation respectively. The optimal values of these design variables represent the optimal solution, or more specifically, the optimal constellation configuration. On the other hand, internal variables are intermediate values derived from operations on design variables. While these variables cannot be controlled directly by the optimization framework, their presence impacts the behavior of the system. Within the context of the constellation design problem, the internal variables are given by $\boldsymbol{y} = [n_t\ \Omega\ \nu]^T$, representing the total number of CubeSats,

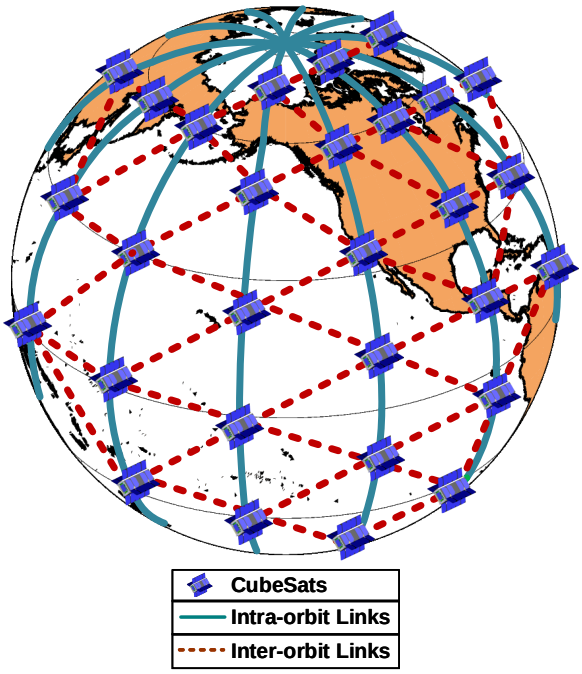


Figure 7.2: Representative CubeSat constellation.

i.e., $n_s n_p$, and the right ascension of the ascending node (RAAN) and true anomaly of the constituent CubeSats. Finally, we have the static parameters whose values remain fixed throughout the SA process. Here $\boldsymbol{z} = [e\ \delta\ d^{RANGE}\ n_t^{MAX}\ \lambda_{UB}\ \lambda_{LB}\ \phi_{UB}\ \phi_{LB}]^T$, represents the orbital eccentricity, minimum elevation angle constraint, maximum feasible ISL range, and maximum number of CubeSats respectively. Furthermore, λ_{UB}, λ_{LB}, ϕ_{UB}, and ϕ_{LB} represent the upper and lower bounds on the longitude and latitudes respectively, defining the region of interest. For example, the parameter set $\lambda_{UB} = 180°$, $\lambda_{LB} = -180°$, $\phi_{UB} = 90°$, and $\phi_{LB} = -90°$, corresponds to the global coverage use case. Since we are dealing with circular orbits exclusively, $e = 0$. Furthermore, unless stated otherwise, distances within the context of this chapter refer to orthodromic distances.

7.3.1 Orbit Propagation Subsystem

Orbit propagation forms the basis of the constellation design framework. The coverage and connectivity estimation modules described in the following two sections rely extensively

[1]Figure 7.1 should not be interpreted as a flowchart.

on orbit data such as the CubSats' SSPs and position vectors, and therefore accurate characterization of the satellites' orbital motion is of vital importance. We begin by defining the orbital elements of a given CubeSat s at time $t = 0$ as $[h_a\ h_p\ e\ \Omega_s(0)\ \omega_s(0)\ \nu_s(0)]^T$, where h_a and h_p represent the apogee and perigee altitude respectively, and $\omega_s(0)$ represents the argument of perigee at time $t = 0$. Given that the orbit is circular, $h_a = h_p = h$, and $\omega_s(0) = 0$. The next task is to determine the values of $\Omega_s(t)$, $\omega_s(t)$, and $\nu_s(t)$ at time t. First, we note that the Earth's oblateness causes small persistent variations of the RAAN and argument of perigee, with their respective rates of change being given by

$$\frac{\mathrm{d}\Omega_s}{\mathrm{d}t} = -\frac{3J_2\cos i}{2}\sqrt{\frac{2\mu}{(2R_E + h_a + h_p)^3}}\left(\frac{2R_E}{(2R_E + h_a + h_p)\,(1-e^2)}\right)^2, \tag{7.1}$$

where μ is the standard Earth gravitational parameter, R_E is the Earth's radius, and J_2 is the spherical harmonic coefficient describing the Earth's oblateness. Furthermore,

$$\frac{\mathrm{d}\omega_s}{\mathrm{d}t} = \frac{3J_2(5\cos^2 i - 1)}{4}\sqrt{\frac{2\mu}{(2R_E + h_a + h_p)^3}}\left(\frac{2R_E}{(2R_E + h_a + h_p)\,(1-e^2)}\right)^2. \tag{7.2}$$

In order to obtain the true anomaly at time t, we first make use of Kepler's equation [178, §4.2] to obtain the mean anomaly, $M_s(0)$, at $t = 0$ as

$$M_s(0) = E_s(0) - e\sin(E_s(0)), \tag{7.3}$$

where $E_s(0)$ is the eccentric anomaly for CubeSat s at time $t = 0$ given by

$$E_s(0) = \arctan2\left(\frac{\sqrt{1-e^2}\sin(\nu_s(0))}{1 + e\cos(\nu_s(0))}, \frac{e + \cos(\nu_s(0))}{1 + e\cos(\nu_s(0))}\right). \tag{7.4}$$

Then, the mean anomaly at time t can be calculated as

$$M_s(t) = M_s(0) + t\sqrt{\frac{2\mu}{(2R_E + h_a + h_p)^3}}, \tag{7.5}$$

followed by the use of (7.3) along with an iterative method, such as Newton's method, to solve for $E_s(t)$. Finally, the true anomaly at time t, $\nu_s(t)$, can be obtained from $E_s(t)$ using (7.4).

Using the trajectory [178, §2.2], we can express the magnitude of the position vector at time t, $r(t)$, for a given CubeSat s as

$$r_s(t) = \frac{a(1-e^2)}{1+e\cos(\nu_s(t))}, \tag{7.6}$$

with the position vector in the perifocal coordinate (PQW) system being given by

$$\boldsymbol{r_s(t)} = \begin{bmatrix} r_s(t)\cos(\nu_s(t)) \\ r_s(t)\sin(\nu_s(t)) \\ 0 \end{bmatrix}, \tag{7.7}$$

followed by a conversion to the Earth-centered inertial (ECI) coordinate frame through the following rotation matrix

$$\left[\frac{\text{ECI}}{\text{PQW}}\right] = R_3 R_2 R_1, \tag{7.8}$$

where

$$R_1 = \begin{bmatrix} \cos(\omega_s(t)) & -\sin(\omega_s(t)) & 0 \\ \sin(\omega_s(t)) & \cos(\omega_s(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{7.9}$$

$$R_2 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & \cos(i) & -\sin(i) \\ 0 & \sin(i) & \cos(i) \end{bmatrix}, \tag{7.10}$$

and

$$R_3 = \begin{bmatrix} \cos(\Omega_s(t)) & -\sin(\Omega_s(t)) & 0 \\ \sin(\Omega_s(t)) & \cos(\Omega_s(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{7.11}$$

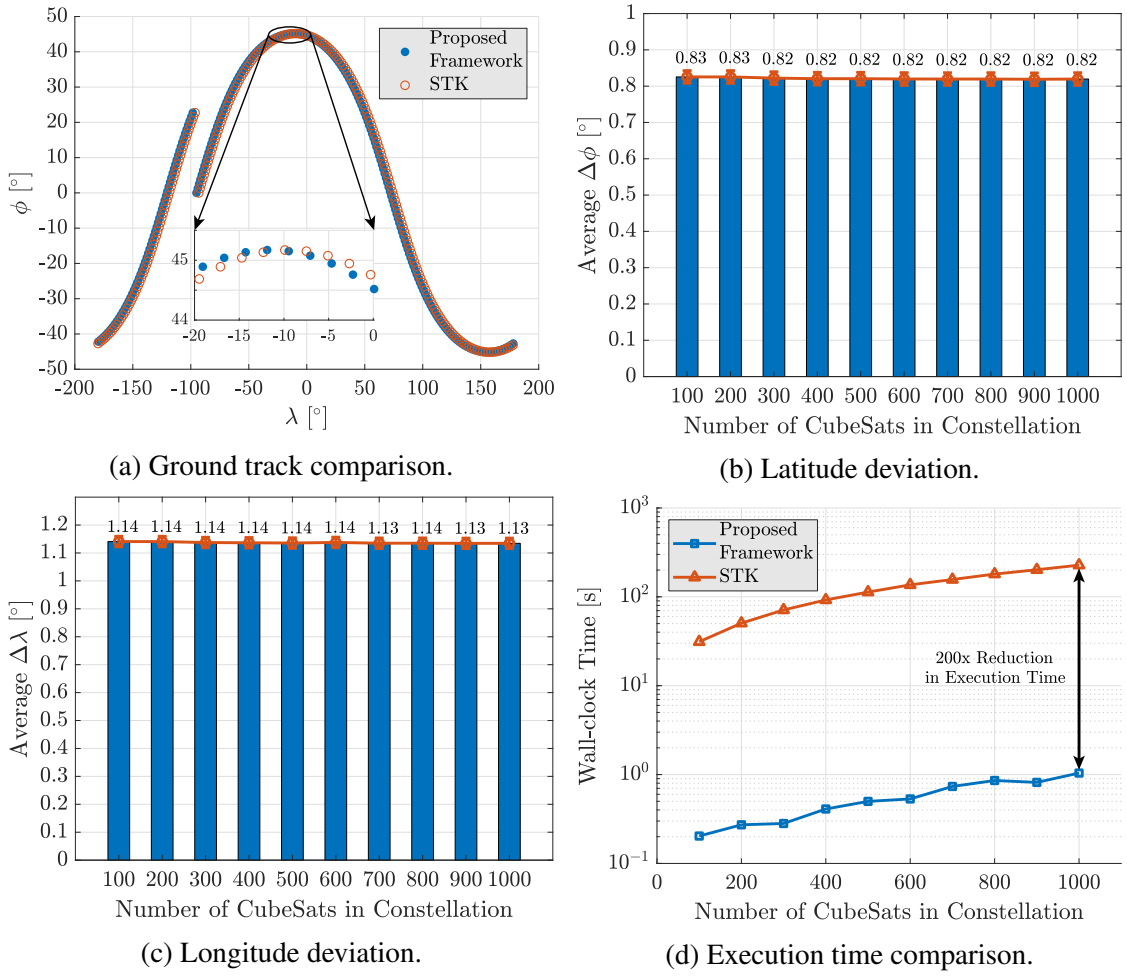
The next step involves transforming the ECI coordinates to the Earth-centered, Earth-fixed (ECEF) coordinate system. In doing so, we have used the J2000 ECI frame of reference [178] along with a modified Julian date-based Greenwich mean sidereal time (GMST) calculation. The corresponding rotation matrix is given by

$$\left[\frac{\text{ECEF}}{\text{ECI}}\right] = \begin{bmatrix} \cos(\omega_E t + \theta(t)) & -\sin(\omega_E t + \theta(t)) & 0 \\ \sin(\omega_E t + \theta(t)) & \cos(\omega_E t + \theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{7.12}$$

where $\theta(t)$ is the GMST for instance t, and $\omega_E = 72.9115\ \mu\text{rad/s}$ is the Earth's rotation rate based on the WGS84 model [179].

Finally, the ECEF value of the position of CubeSat s at time t is converted to the geodetic longitude, $\lambda_s(t)$, and geodetic latitude, $\phi_s(t)$, of the corresponding SSP, $\mathfrak{p}_s(t)$, using the procedure outlined in [180]. In order to verify the accuracy and computing efficiency of the orbit propagation procedure described herein, we compare the results obtained with those from STK's J_2 propagator, which serves as the standard reference, in Figure 7.3. More specifically, we take into a consideration a constellation with circular orbits deployed at an altitude of $780\ \text{km}$ having a uniform distribution of CubeSats with $\omega_s(0) = 0$ and $i = 45°$, and increase the number of CubeSats in the constellation from 100 to 1000 in steps of 100 for a period of $24\ \text{h}$.

First, Figure 7.3a compares the ground track of a single CubeSat, with $\Omega_s(0) = 0$, over its entire orbital period as obtained from our propagator and STK. From the figure it is clear that the propagator described in this section tracks the ground track obtained from STK very closely. Next, in Figure 7.3b and 7.3c, we compare the deviation in the longitude and latitude of the ground track as the number of CubeSats in the constellation increases from 100 to 1000, in order to demonstrate that the accuracy of the propagator is not impacted by the number of CubeSats in the constellation. Here we note that even for the 1000 CubeSat case, the average deviation is around $1°$.



(a) Ground track comparison.

(b) Latitude deviation.

(c) Longitude deviation.

(d) Execution time comparison.

Figure 7.3: Comparison of the proposed orbit propagation subsystem with STK's J2 propagator.

Furthermore, we compare the computation times between the proposed propagator and STK in Figure 7.3d. In the interest of fairness, the results in Figure 7.3d have been obtained by: (i) implementing the presented propagator in MATLAB, and (ii) using STK's MATLAB interface with all graphics and animations disabled for comparison. Furthermore, the run-time comparison presented in the figure has been performed on an AMD workstation with the 3700X processor and 32 GB of RAM. The figure shows the variation in wall-clock time required to obtain the ground track for all CubeSats, over a 24 h period, as the number of CubeSats in the constellation is increased from 100 to 1000. Each data point in Figure 7.3d represents the average run-time obtained after 1000 evaluations.

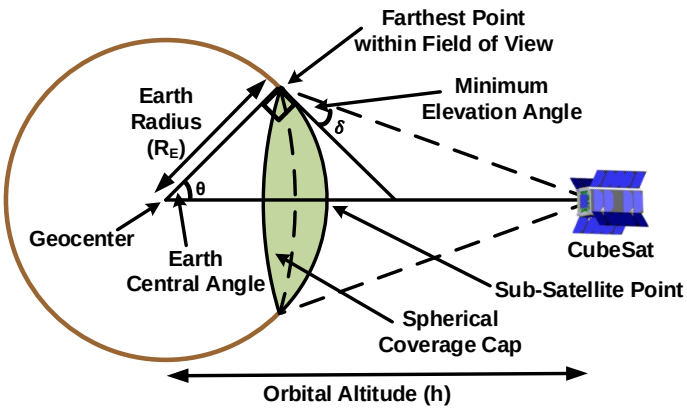


Figure 7.4: CubeSat coverage geometry.

From the figure it is clear that the run-time associated with STK is several orders of magnitude higher than that required by the propagator described herein. In particular, for the 1000 CubeSat case, our orbit propagator is about 200 times faster than STK. Since orbit propagation forms the basis of any constellation design framework, this result is of particular importance because it allows our framework the flexibility of evaluating several constellations at an extremely rapid rate. To this end, we note that our orbit propagator achieves performance that is similar to STK in terms of accuracy even in the worst-case 1000 CubeSat scenario, while requiring significantly lesser computation time.

7.3.2 Coverage Estimation Subsystem

Coverage estimation is a key component of the constellation design framework, with average revisit time, percentage of time covered, average access duration, etc., representing the classical coverage metrics. However, the computation of these metrics is time consuming, and serves as one of the major pitfalls of commercial tools. Instead, we propose an alternate approach based on the concept of spherical Voronoi tessellations, CubeSat revisit frequency, and SSP density. As we will demonstrate in Section 7.5, the method proposed by us can scale well for hundreds, and potentially thousands, of CubeSats.

We begin by defining the spherical radius, d_s^{RAD}, of the spherical cap subtended by CubeSat s on the Earth's surface. Since we are dealing with circular orbits $d_s^{RAD} = d^{RAD}$

$\forall\, s \in \{1,\ 2,\ \ldots,\ n_t\}$, i.e., all CubeSats in the constellation have the same spherical radius, where $d^{RAD} = R_E\theta$. Within this context, R_E is the Earth's radius, and

$$\theta = \pi/2 - \delta - \arcsin\left(\frac{R_E}{R_E + h}\cos\delta\right), \tag{7.13}$$

is the Earth central angle subtended by the CubeSat at the geocenter. The result in (7.13) follows from the application of the law of sines to the triangle formed by the CubeSat, the farthest point within its field of view, and the geocenter, as shown in Figure 7.4, under the assumption that the Earth is a perfect sphere. Furthermore, we note that the set of SSPs at time t, $\mathfrak{p}(t) \coloneqq \{\mathfrak{p}_s(t)\}_{s=1}^{n_t}$, serves as a finite collection of n_t distinct generators over the spherical metric space $\mathbb{X}$ defined by the Earth's surface. It follows that the set

$$\mathfrak{VD}_s(t) \coloneqq \{\mathfrak{p}(t) \in \mathbb{X} \mid \forall s \neq k : dist(\mathfrak{p}(t), \mathfrak{p}_s(t)) \leq dist(\mathfrak{p}(t), \mathfrak{p}_k(t))\}, \tag{7.14}$$

is the spherical Voronoi region associated with SSP $\mathfrak{p}_s(t)$, and thus CubeSat s at time t. The corresponding Voronoi diagram for all n_t CubeSats is thus given by $\mathfrak{VD}(t)$.

Furthermore, in order to better quantify the Earth coverage achieved by the constellation, we introduce the dual of the spherical Voronoi diagram, known as the spherical Delaunay triangulation, $\mathfrak{DT}(t))$. The Delaunay triangulation is obtained by connecting with a line segment any two points $\mathfrak{p}_s(t)$ and $\mathfrak{p}_k(t) \in \mathfrak{p}(t)$ for which a circle $C(t)$ exists that passes through $\mathfrak{p}_s(t)$ and $\mathfrak{p}_k(t)$, and does not contain any other points of $\mathfrak{p}(t)$ in its interior or boundary. Using the representative constellation of Figure 7.2, for some given time t, we are able to obtain the corresponding $\mathfrak{VD}(t)$ and $\mathfrak{DT}(t)$ as shown in Figure 7.5. More generally, the proposed framework uses the STRIPACK algorithm [181] to generate the spherical Voronoi diagram and Delaunay triangulations associated with a set of SSPs.

Based on the Voronoi diagram and Delaunay triangulation concepts, we introduce the following Lemma 1 for coverage characterization.

Lemma 1. *Continuous coverage over a region of interest can be guaranteed iff the radius*

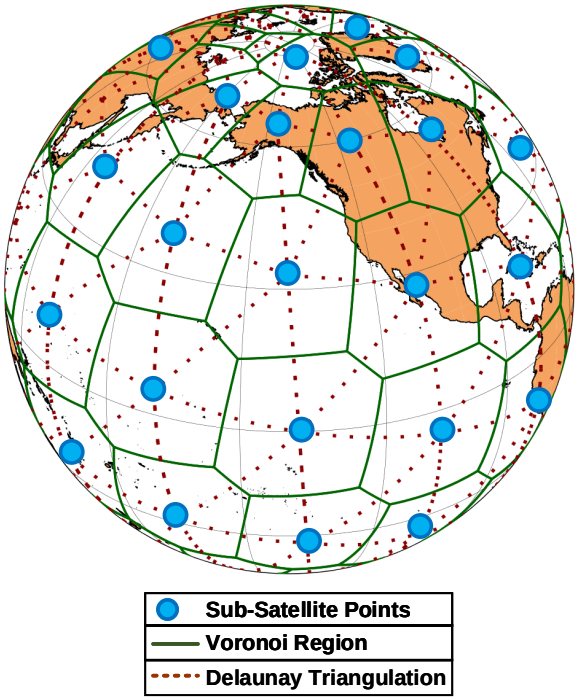


Figure 7.5: Voronoi diagram and Delaunay triangulation formed by the sub-satellite points.

of the spherical cap subtended by any given CubeSat exceeds the maximum radius of the spherical circles that circumscribe the Delaunay triangulation of those SSPs that lie within said region for every time instant under consideration.

Proof. The proof follows from the fact that the circumcenters of Delaunay triangles in $\mathfrak{DT}(t)$ are the vertices of the spherical Voronoi regions in $\mathfrak{VD}(t)$, and the vertices of the Delaunay triangles in $\mathfrak{DT}(t)$ are the generators of the Voronoi regions in $\mathfrak{VD}(t)$. Consequently, the radius of any such circumcircle will be given by the distance between an SSP $\mathfrak{p}_s(t)$ and one of the vertices of the region $\mathfrak{VD}_s(t)$. Therefore, if the radius of the spherical coverage cap exceeds all such distances for the region under consideration, continuous coverage over that region can be guaranteed. Conversely, if the system exhibits continuous coverage, it implies that the radius of any given spherical coverage cap exceeds the maximum possible spherical circle radius within the region of interest. □

The result above can be readily extended to account for global coverage. Thus, we have Corollary 1 for global coverage characterization.

Corollary 1. *For every time instant t, global coverage can be guaranteed iff the radius of every spherical circle that circumscribes the Delaunay triangulation $\mathfrak{DT}(t)$ of the set of SSPs $\mathfrak{p}(t)$ is less than or equal to the radius of the spherical coverage cap d^{RAD}.*

Proof. The result follows from Lemma 1 by expanding the region of interest to include the entire surface of the Earth. □

Algorithm 3 Voronoi Coverage Estimation

for $t \leftarrow 1$ to n_{TIME} **do**
  $f_{COV}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, t) \leftarrow 0$
  $\mathfrak{VD}(t) \leftarrow \text{STRIPACK}(\lambda(t),\ \phi(t))$
  for $s \leftarrow 1$ to n_t **do**
    if $\lambda_{LB} \leq \lambda_s(t) \leq \lambda_{UB}\ \ \&\ \ \phi_{LB} \leq \phi_s(t) \leq \phi_{UB}$ **then**
      $\lambda_{\mathfrak{VD}_s(t)} \leftarrow \lambda_{LB}\ \forall\ \lambda_{\mathfrak{VD}_s(t)} \leq \lambda_{LB}$
      $\lambda_{\mathfrak{VD}_s(t)} \leftarrow \lambda_{UB}\ \forall\ \lambda_{\mathfrak{VD}_s(t)} \geq \lambda_{UB}$
      $\phi_{\mathfrak{VD}_s(t)} \leftarrow \phi_{LB}\ \forall\ \phi_{\mathfrak{VD}_s(t)} \leq \phi_{LB}$
      $\phi_{\mathfrak{VD}_s(t)} \leftarrow \phi_{UB}\ \forall\ \phi_{\mathfrak{VD}_s(t)} \geq \phi_{UB}$
      $d^{MAX} = \max_{1 \leq k \leq |V(\mathfrak{VD}_s(t))|} (distance(s,k))$
      if $d^{RAD} \geq d^{MAX}$ **then**
        $f_{COV}(\cdot) \leftarrow f_{COV}(\cdot) + 1$
      else
        $f_{COV}(\cdot) \leftarrow f_{COV}(\cdot) + \left(d^{RAD}/d^{MAX}\right)^2$
      end if
    end if
  end for
  $f_{COV}(\cdot) \leftarrow f_{COV}(\cdot)/n_t$
end for

Accordingly, we introduce the Voronoi coverage parameter, $f_{COV}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, t)$, which is calculated in accordance with the procedure outlined in Algorithm 3, where $\lambda(t) \coloneqq \{\lambda_s(t)\}_{s=1}^{n_t}$, $\phi(t) \coloneqq \{\phi_s(t)\}_{s=1}^{n_t}$, n_{TIME} refers to the time instances under consideration, and $V(\mathfrak{VD}_s(t))$ represents the set of vertices of the diagram corresponding to CubeSat s at time t. The geodetic longitude and latitude values for these vertices are expressed as $\lambda_{\mathfrak{VD}_s(t)}$ and $\phi_{\mathfrak{VD}_s(t)}$. First, Algorithm 3 uses the STRIPACK subroutine to obtain the Voronoi diagram, $\mathfrak{VD}(t)$, associated with SSPs $\mathfrak{p}(t) = (\lambda(t),\ \phi(t))$. Next, the algorithm filters out non-relevant SSPs by checking whether the SSP for each CubeSat $s \in \{1,\ 2,\ \ldots,\ n_t\}$ lies within

the region of interest set by the constellation designer. For an SSP that lies within the region of interest, the associated Voronoi vertices are adjusted Furthermore, if required, to bring them within the region under consideration. Then, the algorithm calculates the maximum distance between the SSP and its associated Voronoi vertices. If d^{RAD} exceeds this distance, then the region is covered completely, if not, then the proportion of the region under coverage is given by the square of the ratio of the two distances. Once this calculation has been performed for all n_t CubeSats, the result is normalized by the number of CubeSats in the constellation. To summarize, Algorithm 3 realizes Earth coverage as a ratio of the square of the radius of the spherical coverage cap to that of the spherical circles that circumscribe the obtained Delaunay triangulation.

Furthermore, as part of the coverage characterization, we obtain the CubeSat revisit frequency, $f_{REV}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \phi)$, based on the following procedure

$$F_1(\phi) = \begin{cases} \text{sgn}(\phi - \theta), & \text{if } i = 0; \\ \text{sgn}(\pi/2 - \theta - & \\ \qquad \min(i, \pi/2 - i)), & \text{if } \phi = \pi/2; \\ \frac{-\sin\theta + \sin\phi\cos i}{\cos\phi \sin i}, & \text{otherwise,} \end{cases} \tag{7.15}$$

$$F_2(\phi) = \begin{cases} 1, & \text{if } i = 0 \text{ or } \phi = \pi/2; \\ \frac{\sin\theta + \sin\phi\cos i}{\cos\phi\sin i}, & \text{otherwise,} \end{cases} \tag{7.16}$$

$$F(\phi) = \frac{1}{\pi}\left(\arccos\{\min[1, \max(-1, F_1(\phi))]\} - \arccos\min[1, F_2(\phi)]\right), \tag{7.17}$$

$$F_{REV}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \phi) = \left(1 - \frac{2\pi\sqrt{(R_E + h)^3/\mu}}{T}\cos i\right) F(\phi), \tag{7.18}$$

where T is the duration of one solar day. Additional details concerning the parameters $F_1(\phi)$ and $F_2(\phi)$ can be found in [182], and have been omitted here for sake of brevity. A

combination of $f_{COV}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, t)$ and $f_{REV}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \phi)$ works well in practice for the purpose of coverage characterization. The former quantifies Earth coverage in terms of surface area, while the latter describes coverage in terms of CubeSat availability. In addition to the aforementioned analytical coverage proof, we will also use the proposed coverage metrics, i.e., $f_{COV}(\cdot)$ and $f_{REV}(\cdot)$, to design a coverage-optimized constellation in this section, to demonstrate the practicality of the proposed metrics.

Simultaneously, we note that $f_{REV}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \phi)$ is a latitude-centric metric, and therefore well-suited for use cases where the focus is on either global or latitude-specific coverage. However, $f_{REV}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \phi)$ cannot be used when designing constellations for use cases that require region-specific coverage, since it only provides us with information about the revisit statistics associated with a given latitude, as opposed to a specific geodetic point.

Consequently, we introduce $f_{DEN}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, t)$ for region-specific scenarios, representing the SSP density within the region of interest at time t. In doing so, we first define indicator variable $\alpha_s(t) \; \forall \; s \in \; \{1, \; 2, \; \ldots, \; n_t\}, \; t \in \{1, \; 2, \; \ldots, \; n_{TIME}\}$ such that

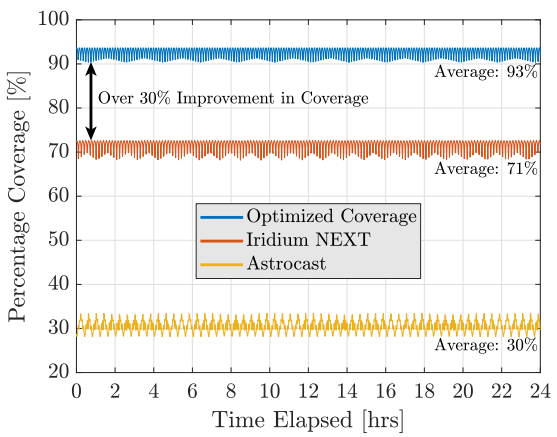
$$\alpha_s(t) = \begin{cases} 1, \text{ if } \lambda_{LB} \leq \lambda_s(t) \leq \lambda_{UB}, \text{ and} \\ \qquad \phi_{LB} \leq \phi_s(t) \leq \phi_{UB}; \\ 0, \text{ otherwise,} \end{cases} \tag{7.19}$$

then, $f_{DEN}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, t)$ can be obtained as

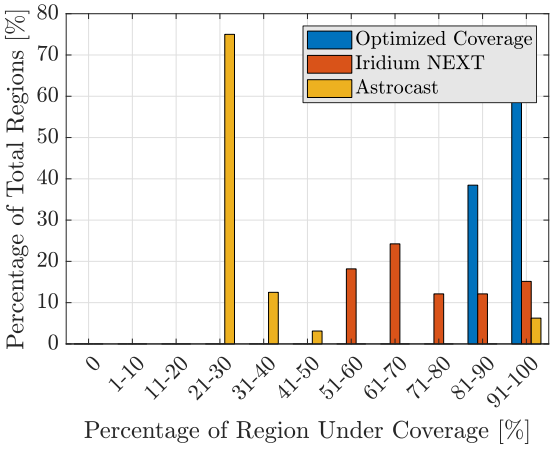
$$f_{DEN}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, t) = \frac{\sum_{s=1}^{n_t} \alpha_s(t)}{n_t}. \tag{7.20}$$

For region-specific coverage scenarios, $f_{DEN}(\cdot)$ works well in tandem with $f_{COV}(\cdot)$, because while the former tracks the number of CubeSats that are within the region of interest over time, the latter describes the coverage of the CubeSats under consideration.

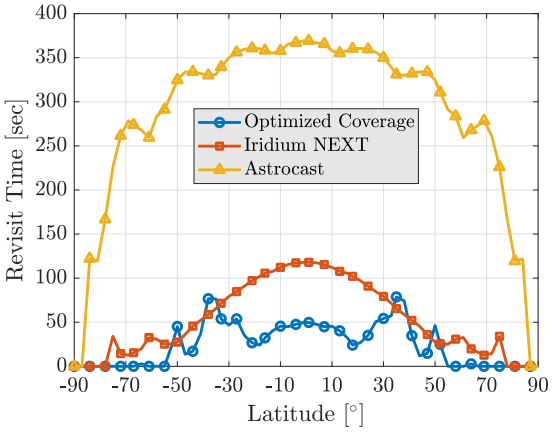
While Lemma 1 provides an analytical justification regarding the efficacy of the coverage estimation subsystem, in order to further reinforce its validity, we demonstrate the



(a) Variation in percentage coverage with time.



(b) Percentage coverage by region.



(c) Revisit time by latitude.

Figure 7.6: Comparison of coverage-related metrics.

impact of optimizing for coverage using a small-scale constellation. The primary idea here is to compare the coverage metrics of a constellation that is obtained by leveraging the proposed coverage estimation subsystem, against other similarly sized constellations that target global coverage. More specifically, using the framework described in Section 7.4 with weights $w_1 = w_4 = 0$, we attempt to obtain a candidate constellation, not exceeding 80 CubeSats, that has been designed for optimal coverage. For the purpose of comparison, we have chosen the Iridium NEXT and Astrocast constellations, since both aim to achieve global coverage, and contain a similar number of satellites, at 66 and 64 respectively. Our design optimization framework provides the following coverage-optimized constellation configuration: $[899.674\ 84.664\ 13\ 6\ 1]^T$, i.e., a total of 78 CubeSats arranged in 6 orbits of

13 CubeSats each, at an altitude of $899.674\ \mathrm{km}$ with an inclination of $84.664°$. A cursory examination of the obtained constellation parameters suggests that our framework proposes a marginal increases in the number of satellites, from 66 to 78, and a 15% increase in the orbital altitude from $780\ \mathrm{km}$ to $899\ \mathrm{km}$, when compared to Iridium NEXT.

In order to quantify the impact of this change on constellation performance, we compare coverage-related metrics, such as percentage coverage and revisit time, with those of Iridium NEXT and Astrocast, as shown in Figure 7.6. From Figure 7.6a we note that the coverage optimized constellation provides a significant 30% improvement in coverage over the Iridium NEXT constellation, achieving over 90% Earth coverage over the $24\ \mathrm{h}$ observation period. Of particular note is the result showcased in Figure 7.6b, wherein it has been demonstrated that with the coverage optimized constellation, any given region has a minimum of at least 81% coverage, a substantial improvement over the 51% offered by Iridium, and the 21% offered by Astrocast. Furthermore, we also compare the revisit time metrics across all three constellations, with the intention of the demonstrating that the coverage metrics proposed by us lead to a marked improvement in the revisit time performance of the constellation. The results in Figure 7.6c have been obtained using STK, and we note that the optimized coverage constellation offers a significant improvement over both Astrocast and Iridium NEXT, achieving less than one-sixth and one-half of their respective average revisit times.

In this manner, we have demonstrated the efficacy of the proposed coverage estimation subsystem of our framework. In particular, the use of the coverage estimation subsystem in designing constellations leads to a guaranteed improvement in coverage performance.

7.3.3 Connectivity Estimation Subsystem

Connectivity estimation is vital from a data communications perspective. Within the context of system design, we are interested in the ISL density of the constellation, i.e., the ratio of the number of active ISLs at any given point in time, to the ideal number of ISLs as

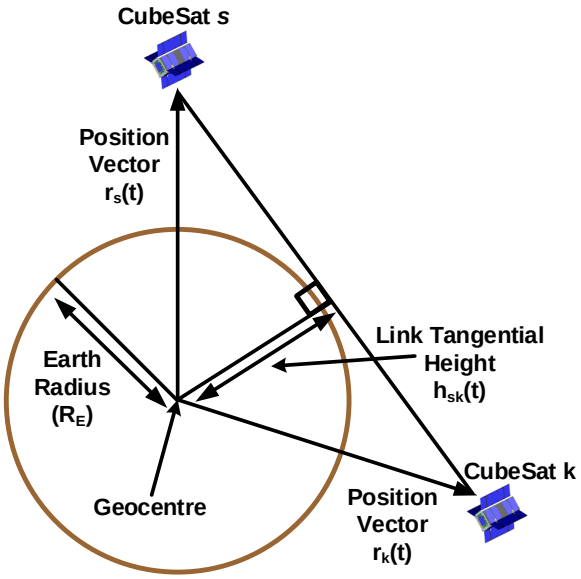


Figure 7.7: Inter-satellite link visibility.

defined in the system specifications. The higher is the ISL density, the more robust will be the connectivity. Expectedly, a higher ISL density requires a larger number of CubeSats and a longer communication range. However, from a system optimization perspective, we are trying to minimize the number of satellites in the constellation, and therefore a trade-off exists between connectivity and the number of CubeSats. As part of the connectivity estimation subsystem, we first determine the maximum number of ISLs in the constellation at time t, $n_{ISL}^{MAX}(t)$. In doing so, we introduce the variable $n_{REG}(t) = \sum_{s=1}^{n_t} \alpha_s(t) \; \forall \; t \in \{1, \ldots, n_{TIME}\}$ which represents the number of CubeSats with SSPs that lie within the region of interest, with

$$n_{ISL}^{MAX}(t) = \frac{n_{REG}(t)\,(n_{REG}(t) - 1)}{2}. \tag{7.21}$$

However, it is not practically feasible to have $n_{ISL}^{MAX}(t)$ ISLs active at all times, due to factors that impact ISL availability such as CubeSat visibility and communication range. Instead, the ideal number of ISLs at time t is expressed as $n_{ISL}(t) = \beta n_{ISL}^{MAX}(t)$, where $0 \leq \beta \leq 1$ is the constellation connectivity parameter. We leave the choice of β to the system designer, depending on the kind of connectivity required. Furthermore, in order to

quantify CubeSat visibility for any two CubeSats s and k, with $s \neq k$, we note that if the angle between their respective position vectors, $\boldsymbol{r_i(t)}$ and $\boldsymbol{r_j(t)}$, is an acute angle, then the two CubeSats are visible to each other. On the other hand, as shown in Figure 7.7, if this angle is obtuse, then the associated link tangential height $h_{ij}(t)$, must be at least greater than the radius of the Earth in order to ensure visibility. More formally, we introduce indicator variable $\gamma_{sk}(t) \; \forall \; s, \; k \in \{1, \; 2, \; \ldots, \; n_t\}, \; t \in \{1, \; 2, \; \ldots, \; n_{TIME}\}$ such that

$$\gamma_{sk}(t) = \begin{cases} 1, \text{ if } \boldsymbol{r_s(t)} \cdot \boldsymbol{r_k(t)} \geq 0; \\ 1, \text{ if } \boldsymbol{r_s(t)} \cdot \boldsymbol{r_k(t)} < 0, \text{ and} \\ \quad h_{sk}(t) = \frac{|\boldsymbol{r_s(t)} \times \boldsymbol{r_k(t)}|}{|\boldsymbol{r_s(t)} - \boldsymbol{r_k(t)}|} \geq R_E; \\ 0, \text{ otherwise.} \end{cases} \tag{7.22}$$

The second parameter that influences ISL feasibility is the distance between the CubeSats, i.e., if the distance between any two CubeSats exceeds the maximum communication range, a link cannot be established. Accordingly, we introduce the indicator variable $d_{sk}(t) \; \forall \; s, \; k \in \{1, \; 2, \; \ldots, \; n_t\}, \; t \in \{1, \; 2, \; \ldots, \; n_{TIME}\}$ as follows

$$d_{sk}(t) = \begin{cases} 1, \text{ if } dist(i,j) \leq d^{RANGE} \text{ at time } t; \\ 0, \text{ otherwise.} \end{cases} \tag{7.23}$$

We note that the value of d^{RANGE} is a function of the link budget, and thus depends upon a variety of factors ranging from the transmit power and frequency bands in use, to the channel model and the antenna design of the CubeSat. With a view to keeping the proposed framework flexible, we prefer to use the generic parameter d^{RANGE}, instead of specifying the link budget. The idea here is that the system designer can calculate the radio communication range specific to their scenario and simply substitute the appropriate value for d^{RANGE}. Furthermore, with $\gamma_{sk}(t)$ and $d_{sk}(t)$ defined, we utilize Algorithm 4 to obtain the connectivity metric $f_{ISL}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, t)$, which represents the ratio of the active ISLs to the ideal number of ISLs at time t. In particular, for every time instance t, for every CubeSat s that lies within

the region of interest, Algorithm 4 checks the connectivity with every other CubeSat k based on: (i) the target Cubesat's location ($\alpha_k(t)$), (ii) the target CubeSat's visibility ($\gamma_{sk}(t)$), and (iii) the distance to the target CubeSat ($d_{sk}(t)$). The number of ISLs thus obtained are then normalized with respect to the ideal number of ISLs at time t, $n_{ISL}(t)$.

Algorithm 4 ISL Connectivity Estimation

```
for t ← 1 to n_TIME do
    f_ISL(x, y, z, t) ← 0
    if n_ISL(t) > 0 then
        for s ← 1 to n_t − 1 do
            if α_s(t) = 1 then
                for j ← i + 1 to n_t do
                    if α_k(t)γ_sk(t)d_sk(t) = 1 then
                        f_ISL(·) ← f_ISL(·) + 1
                    end if
                end for
            end if
        end for
        f_ISL(·) ← f_ISL(·)/n_ISL(t)
    end if
end for
```

7.4 Design Optimization Framework

Having described the key subsystems in the preceding sections, we now turn our attention to the large-scale constellation design problem formulation. First, in Section 7.4.1, we formulate the design problem, and then in Section 7.4.2, we present the SA-based solution framework.

7.4.1 Problem Formulation

We note that the static parameters $\boldsymbol{z} = [e\ \delta\ d^{RANGE}\ n_t^{MAX}\ \lambda_{UB}\ \lambda_{LB}\ \phi_{UB}\ \phi_{LB}]^T$ serve as input to the system. Since the framework is specifically concerned with circular orbits, as described in Section 7.3, $e = 0$. Given input $\boldsymbol{z}$, and the coverage and connectivity related metrics from the respective subsystems, we need to determine the optimal value of

the design variables, $\boldsymbol{x^*} = [h^*\ i^*\ n_s^*\ n_p^*\ f_p^*]^T$, which will characterize the optimal constellation configuration. In order to so, we need to define the objective function $G(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, which takes into consideration: (i) the number of CubeSats in the constellation, (ii) the two coverage metrics, and (iii) the connectivity metric.

First, we introduce $G_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ to reflect the constellation density as follows

$$G_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \frac{n_s n_p}{n_t^{MAX}}. \tag{7.24}$$

Constellation density is an important metric as it helps control the number of CubeSats in the constellation. Next, we consider the Voronoi coverage metric, $G_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, as

$$G_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \left(1 - \frac{\sum_{t=1}^{n_{TIME}} w_{COV}(t) f_{COV}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, t)}{n_{TIME}}\right), \tag{7.25}$$

where n_{TIME} refers to the time steps under consideration. The weights $0 \leq w_{COV}(t) \leq 1\ \forall\ t \in \{1,\ 2,\ \ldots,\ n_{TIME}\}$, with $0 \leq \sum_t w_{COV}(t) \leq 1$, allow us to tune the coverage for specific times of the day, if required. Furthermore, if the use case under consideration requires either global or latitude-specific coverage, the revisit metric, $G_3^G(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, is given by

$$G_3^G(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \left(1 - \frac{\sum_{\phi=-90^\circ}^{90^\circ} w_{REV}(\phi) f_{REV}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \phi)}{n_{LAT}}\right), \tag{7.26}$$

where $n_{LAT} = 181$ refers to the number of latitudes under consideration, and $0 \leq w_{REV}(\phi) \leq 1\ \forall\ \phi \in [-90\ 90]$, with $0 \leq \sum_\phi w_{REV}(\phi) \leq 1$. On the other hand, for regional coverage requirements, where $f_{REV}(\cdot)$ is replaced by the SSP density parameter, we have

$$G_3^R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \left(1 - \frac{\sum_{t=1}^{n_{TIME}} w_{DEN}(t) f_{DEN}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, t)}{n_{TIME}}\right), \tag{7.27}$$

with $0 \leq w_{DEN}(t) \leq 1\ \forall\ t \in \{1,\ 2,\ \ldots,\ n_{TIME}\}$, and $0 \leq \sum_t w_{DEN}(t) \leq 1$. Since $G_3(\cdot)$ can either take value $G_3^G(\cdot)$ or $G_3^R(\cdot)$, but not both for the same coverage scenario, we

introduce an indicator variable g such that

$$g = \begin{cases} 1, \text{ if } \lambda_{LB} = -180^\circ \text{ and } \lambda_{UB} = 180^\circ; \\ 0, \text{ otherwise,} \end{cases} \tag{7.28}$$

with $G_3(\cdot) = (1-g)G_3^G(\cdot) + gG_3^R(\cdot)$. Finally, we have the ISL connectivity metric, $G_4(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, as

$$G_4(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \left(1 - \frac{\sum_{t=1}^{n_{TIME}} w_{ISL}(t) F_{ISL}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, t)}{n_{TIME}}\right), \tag{7.29}$$

with $0 \leq w_{ISL}(t) \leq 1 \; \forall \; t \in \{1,\ 2,\ \ldots,\ n_{TIME}\}$, and $0 \leq \sum_t w_{ISL}(t) \leq 1$. The objective function can then be expressed as

$$G(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = w_1 G_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) + w_2 G_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) + w_3 G_3(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) + w_4 G_4(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}). \tag{7.30}$$

We have chosen a weighted summation of the different metrics in order to obtain a single objective in the interest of computational speed. The weights w_1, w_2, w_3, and w_4, are intended to allow for flexibility in the prioritization of different metrics, with $0 \leq w_1,\ w_2,\ w_3,\ w_4 \leq 1$, and $0 \leq w_1 + w_2 + w_3 + w_4 \leq 1$. If required, the system designer can choose to optimize for each metric individually by setting the other weights to 0. Furthermore, we note that $\eta = \frac{1}{G(\cdot)}$ represents the constellation quality metric which is a focal point of the performance comparison in Section 7.5.

Next, in order to define the problem constraints, we note that the minimum number of satellites per orbital plane and minimum number of orbital planes can be obtained as $n_s^{MIN} = \lceil 2\pi/(2\theta) \rceil$ and $n_p^{MIN} = \lceil 2\pi/(4\theta) \rceil$ respectively [143]. Additionally, we also provision for constraints on the orbital altitude and inclination, with h^{MAX} and h^{MIN} representing the bounds on the orbital altitude, and i^{MAX} and i^{MIN} quantifying the bounds on the inclination. With this, the constellation design problem can be defined as follows.

Definition 1. ***Large-scale Constellation Design Problem (LsCD).***

Given a set of static parameters of the form $\boldsymbol{z} = [e\ \delta\ d^{RANGE}\ n_t^{MAX}\ \lambda_{UB}\ \lambda_{LB}\ \phi_{UB}\ \phi_{LB}]^T$*, use the internal variables* $\boldsymbol{y} = [n_t\ \Omega\ \nu]^T$*, and system metrics* $f_{COV}(\cdot)$*,* $f_{REV}(\cdot)$*,* $f_{DEN}(\cdot)$*, and* $f_{ISL}(\cdot)$ *to determine the optimal large-scale constellation configuration given by* $\boldsymbol{x}^* = [h^*\ i^*\ n_s^*\ n_p^*\ f_p^*]^T$*. More formally,*

$$
\begin{aligned}
&\textbf{Given:} && \boldsymbol{z} = [e\ \delta\ d^{RANGE}\ n_t^{MAX}\ \lambda_{UB}\ \lambda_{LB}\ \phi_{UB}\ \phi_{LB}]^T,\ n_s^{MIN},\ n_p^{MIN} \\
&\textbf{Find:} && \boldsymbol{x} = [h\ i\ n_s\ n_p\ f_p]^T \\
&\textbf{Minimize:} && G(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \\
&\textbf{Subject to} && \begin{cases} 0 \leq w_1 + w_2 + w_3 + w_4 \leq 1, \\ h^{MIN} \leq h \leq h^{MAX}, \\ i^{MIN} \leq i \leq i^{MAX}, \\ n_s^{MIN} \leq n_s, \\ n_p^{MIN} \leq n_p, \\ 0 \leq f_p \leq n_p - 1, \\ n_t \leq n_t^{MAX}, \end{cases}
\end{aligned}
$$

where $0 \leq w_1,\ w_2,\ w_3,\ w_4 \leq 1$, $h, i \in \mathbb{R}$, and n_s, n_p, $f_p \in \mathbb{Z}$. LsCD represents a non-linear combinatorial optimization problem with a complicated structure that does not admit an easy solution. To this end, we leverage meta-heuristic methods for solving the LsCD problem. In particular, we take into consideration simulated annealing (SA) [183], [184, §2.1], which is a single-state method, and genetic algorithms (GA) [185], which belong to the class of meta-heuristics known as population methods. In comparing the two, we note that a variant of SA known as Adaptive Simulated Annealing (ASA) [186] is demonstrably more efficient than GA-based methods [187]. In particular, for a problem dimension equal to that of LsCD, it has been shown that ASA is an order of magnitude faster than GA [187]. This result is of increased significance when we consider the size of the LsCD solution space. Therefore, we solve LsCD using the ASA method.

7.4.2 Simulated Annealing

The SA algorithm draws upon the analogy between the annealing process used in metallurgy and the problem of solving large combinatorial optimization programs. Within the context of metallurgy, annealing denotes the physical process in which a solid material, placed in a heat bath, is heated by increasing the temperature of the heath bath to a certain maximum value. At this stage, the solid is in a liquid state, characterized by the random arrangement of its particles. The material is then cooled slowly by lowering the temperature of the heat bath. As the temperature is lowered and the material cools down, the constituent particles arrange themselves in a state of minimal internal energy, provided that the maximum temperature is sufficiently high, and that the cooling is carried out sufficiently slowly. If care is not taken to control the cooling rate, the solid will settle down in a meta-stable state with non-minimal internal energy. Thus, the system temperature controls the state transitions. At higher temperatures the material can freely transition to states of higher energy, allowing the system to escape locally minimal energy states, and move towards the global minimum state. However, as the temperature is lowered, the material can only transition to increasingly lower energy states, eventually settling down in the minimal energy state.

Algorithmically, at each annealing temperature Γ_k, a new solution $\boldsymbol{x}_{k+1}$, i.e., a new state, is generated at random based on an annealing function, and the acceptance of this state is based on the Metropolis criterion [188] and an acceptance probability $\mathbb{P}(\boldsymbol{x}_{k+1})$. More specifically, if the change in the objective function, $G(\boldsymbol{x_{k+1}}, \boldsymbol{y}_{k+1}, \boldsymbol{z}) - G(\boldsymbol{x_k}, \boldsymbol{y}_k, \boldsymbol{z})$, is negative, then it implies that $\boldsymbol{x}_{k+1}$ is a better solution than $\boldsymbol{x}_k$ and should be accepted as such. However, if this difference is positive, then $\boldsymbol{x}_{k+1}$ is a worse solution, and its acceptance depends upon $\mathbb{P}(\boldsymbol{x}_{k+1}) \leq r$, where $r \in \mathbb{U}(0, 1)$. At higher temperatures, the system can freely move from a lower energy state, i.e., lower objective value, to a higher energy state, allowing the system to effectively escape local minima. Eventually, as the temperature is reduced in accordance with a cooling schedule, these state transitions reduce, and the system settles down to a minimum energy state, signifying that an approximate solution to

the problem has been found.

Within the context of the LsCD problem, we note that conventional annealing functions are not applicable for two reasons: (i) the upper bound on the design variable f_p is not static, and instead depends on n_p, and (ii) there exists an upper bound on the internal variable n_t, of the form $n_t \leq n_t^{MAX}$. Therefore, we make use of a custom annealing function outlined in Algorithm 5, which ensures that the generated solution, $\boldsymbol{x}_{k+1}$, is always feasible. Here $\boldsymbol{x}_{LB}$ and $\boldsymbol{x}_{UB}$ represent the lower and upper bounds defined previously, and t_k, α_k, $\beta_k \in \mathbb{U}(0,1)$.

More specifically, for every candidate solution, $\boldsymbol{x}_{k+1}$, Algorithm 5 tracks the total number of CubeSats, $n_{t(k+1)}$. In the event that $n_{t(k+1)}$ exceeds n_t^{MAX}, the algorithm then compares the number of CubeSats per orbit, $n_{s(k+1)}$, and the number of orbital planes, $n_{p(k+1)}$, in the constellation. For example, if $n_{s(k+1)}$ exceeds $n_{p(k+1)}$, $n_{s(k+1)}$ is recalculated as $\max(\lfloor n_t^{MAX} n_{s(k+1)}/n_{t(k+1)} \rfloor, n_s^{MIN})$, with $n_{p(k+1)}$ being set to $\max(\lfloor n_t^{MAX}/n_{s(k+1)} \rfloor, n_p^{MIN})$. Doing so ensures that $n_{t(k+1)}$ does not exceed n_t^{MAX} at any point. At the same time, the algorithm takes a weighted summation of the values thus obtained with the corresponding values from the previous iteration, such that the new design values are not too far off from the previous iteration. Furthermore, as is standard practice, all five components of the solution vector $\boldsymbol{x}_{k+1}$ are compared against their respective bounds and adjusted accordingly, followed by a projection of the solution onto the feasible region. Finally, for the global coverage use case, the phasing parameter, $f_{p(k+1)}$, is set to 1, in line with the other state-of-the-art global coverage constellations. For all other use cases, if the phasing parameter happens to exceed the number of orbital planes in the constellation, $f_{p(k+1)}$ is scaled down. Additionally, the acceptance probability, $\mathbb{P}(\boldsymbol{x}_{k+1})$, is given by

$$\mathbb{P}(\boldsymbol{x}_{k+1}) = \frac{1}{1 + \exp\left(\frac{G(\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1}, \boldsymbol{z}) - G(\boldsymbol{x}_k, \boldsymbol{y}_k, \boldsymbol{z})}{\Gamma_k}\right)}. \tag{7.31}$$

We have mentioned the importance of the cooling schedule previously, and note that the

exponential cooling schedule works best with LsCD problem. Therefore,

$$\Gamma_k = 0.95^k \Gamma_0, \tag{7.32}$$

where Γ_0 is the initial temperature. With this, our description of the SA-based optimization framework is complete, and we proceed to discuss and analyze the results obtained in the next section.

Algorithm 5 LsCD Annealing Function

$\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + t_k \Gamma_k$
$n_{t(k+1)} \leftarrow n_{s(k+1)} n_{p(k+1)}$
while $n_{t(k+1)} > n_t^{MAX}$ **do**
  if $n_{s(k+1)} > n_{p(k+1)}$ **then**
    $n_{s(k+1)} \leftarrow \max(\lfloor n_t^{MAX} n_{s(k+1)} / n_{t(k+1)} \rfloor, n_s^{MIN})$
    $n_{p(k+1)} \leftarrow \max(\lfloor n_t^{MAX} / n_{s(k+1)} \rfloor, n_p^{MIN})$
  else
    $n_{p(k+1)} \leftarrow \max(\lfloor n_t^{MAX} n_{p(k+1)} / n_{t(k+1)} \rfloor, n_p^{MIN})$
    $n_{s(k+1)} \leftarrow \max(\lfloor n_t^{MAX} / n_{p(k+1)} \rfloor, n_s^{MIN})$
  end if
  $n_{s(k+1)} \leftarrow \alpha_k n_{s(k+1)} + (1 - \alpha_k) n_{s(k)}$
  $n_{p(k+1)} \leftarrow \alpha_k n_{p(k+1)} + (1 - \alpha_k) n_{p(k)}$
end while
for $i \leftarrow 1$ to 5 **do**
  if $\boldsymbol{x}_{k+1}[i] < \boldsymbol{x}_{LB}[i]$ **then**
    $\boldsymbol{x}_{k+1}[i] \leftarrow \boldsymbol{x}_{LB}[i]$
  else if $\boldsymbol{x}_{k+1}[i] > \boldsymbol{x}_{UB}[i]$ **then**
    $\boldsymbol{x}_{k+1}[i] \leftarrow \boldsymbol{x}_{UB}[i]$
  end if
  $\boldsymbol{x}_{k+1}[i] \leftarrow \beta_k \boldsymbol{x}_{k+1}[i] + (1 - \beta_k) \boldsymbol{x}_k[i]$
end for
$n_{s(k+1)} \leftarrow \lfloor n_{s(k+1)} \rfloor$
$n_{p(k+1)} \leftarrow \lfloor n_{p(k+1)} \rfloor$
if $\lambda_{UB} = 180 \ \ \& \ \ \lambda_{LB} = -180$ **then**
  $f_{p(k+1)} \leftarrow 1$
else
  if $f_{p(k+1)} > n_{p(k+1)} - 1$ **then**
    $f_{p(k+1)} \leftarrow n_{p(k+1)} - 1$
  end if
end if
return $\boldsymbol{x}_{k+1}$

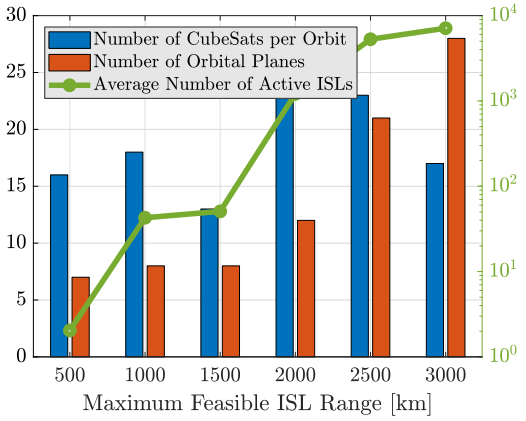
Table 7.2: IoST constellation design configurations for global coverage

Maximum ISL Range (d^{RANGE})	**Optimal IoST Constellation** ($\boldsymbol{x^*} = [h^*\ i^*\ n_s^*\ n_p^*\ f_p^*]^T$)
500 km	$[862.49\ 78.35\ 16\ 7\ 1]^T$
1000 km	$[754.57\ 77.47\ 18\ 8\ 1]^T$
1500 km	$[899.55\ 78.59\ 13\ 8\ 1]^T$
2000 km	$[724.68\ 76.11\ 23\ 12\ 1]^T$
2500 km	$[723.27\ 77.98\ 23\ 21\ 1]^T$
3000 km	$[759.20\ 76.10\ 17\ 28\ 1]^T$

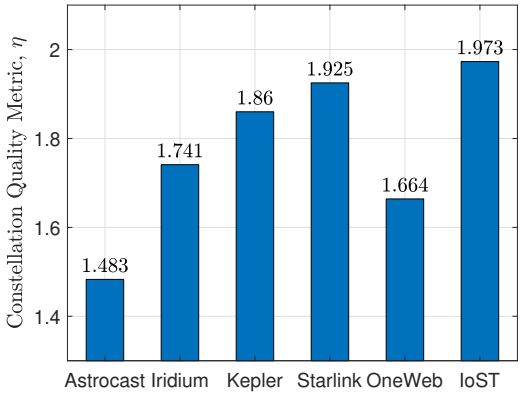
7.5 Constellation Designs for IoST

Having presented the details concerning the design optimization framework, in this section, we use a MATLAB-based implementation of the framework to design constellations for IoST pertaining to three distinct use case requirements: (i) global coverage, (ii) latitude-specific coverage, and (iii) regional coverage. For example, the global coverage use case is well-suited for applications such as worldwide connectivity, while the regional coverage use case serves applications such as localized terrain monitoring and reconnaissance. For the global coverage case, we compare the constellation, as designed by the presented framework, with several state-of-the-art global constellations. Furthermore, for each coverage use case, we vary different elements of the static parameter $\boldsymbol{z}$ to examine the impact of system parameter variation on constellation design.

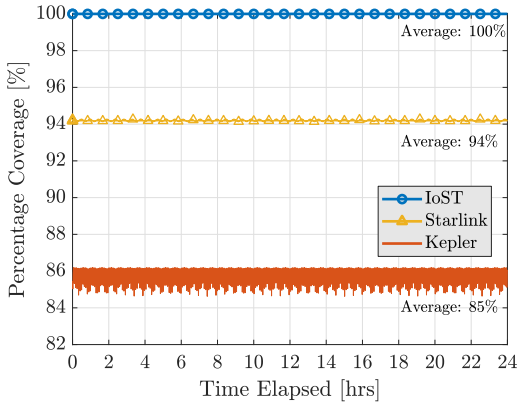
We note that $e = 0$, $n_t^{MAX} = 500$, and $\Gamma_0 = 100^\circ$ C throughout, with $w_1 = 0.17$, $w_2 = 0.25$, $w_3 = 0.25$, and $w_4 = 0.33$, wherein Γ_0 represents the standard initial temperature for SA. Furthermore, as described previously, IoST is meant to be deployed in the exosphere at altitudes between 600 km and 900 km, therefore $h^{MIN} = 600$ km and $h^{MAX} = 900$ km. Additionally, we restrict the design to prograde orbits only with inclinations that do not exceed 90°. In particular, our choice of weights w_1 through w_4 is motivated by the relative



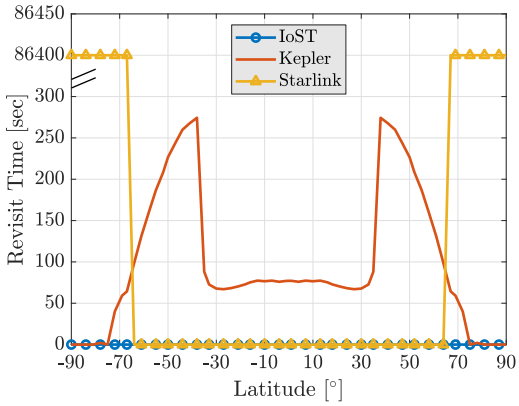
(a) Variation in constellation parameters with change in feasible ISL range.



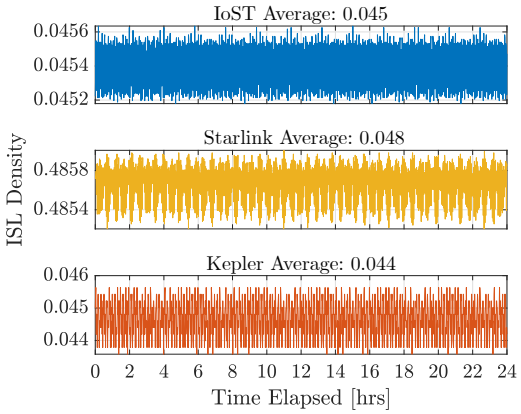
(b) Performance comparison of the IoST constellation with the state-of-the-art solutions.



(c) Percentage coverage comparison between IoST, Starlink, and Kepler.



(d) Revisit time comparison between IoST, Starlink, and Kepler.



(e) ISL density comparison between IoST, Starlink, and Kepler.

Figure 7.8: The global coverage use case.

importance of connectivity within IoST, therefore, we accord it a higher priority than other metrics. On the other hand, the weights associated with the two coverage related metrics take the same value, reflecting their equal relative importance. In general, constellation designers may set these weights according to the specific use case they wish to optimize for.

7.5.1 The Global Coverage Use Case

The global coverage use case is characterized by $\lambda_{UB} = 180°$, $\lambda_{LB} = -180°$, $\phi_{UB} = 90°$, and $\phi_{LB} = -90°$. First, for a minimum elevation angle of $\delta = 15°$, we vary the maximum feasible ISL range, d^{RANGE}, from 500 km to 3000 km in steps of 500 km, and the constellations thus obtained from the framework have been listed in Table 7.2. We have selected $\delta = 15°$ in accordance with the existing literature associated with CubeSats [41, 189, 190]. Then, we compare the constellation obtained for $\delta = 15°$ and 2500 km with existing state-of-the art solutions, such as Astrocast, Iridium NEXT, Kepler, Starlink, and OneWeb. The resulting comparison along with the data in Table 7.2 has been summarized in Figure 7.8.

First, Figure 7.8a characterizes the change in the number of CubeSats per orbit and the number of orbits as d^{RANGE} is varied. The primary motivation for varying d^{RANGE} comes from the fact that it is closely tied to the communication subsystem of the constituent CubeSats. For example, the system designer can modify metrics such as transmit power and the transmit and receive gains, in order to obtain the desired d^{RANGE} value. Therefore, varying d^{RANGE} allows us to examine the impact varying levels of connectivity have on the design of the constellation.

From the figure we note that initially as d^{RANGE} is increased from 500 km to 1500 km the number of CubeSats in the constellation does not change significantly, staying at around 120 CubeSats on an average. However, on increasing d^{RANGE} to 2000 km, and then to 2500 km, the number of CubeSats steadily increases from 276 to 483. While this result might feel counterintuitive at first, the trend is best explained by examining the number of

active ISLs in each case. For relatively short ISL ranges, increasing the number of CubeSats does not have a significant impact on connectivity since the link distance serves as the limiting factor. For example, for a 500 CubeSat constellation, the average inter-satellite distance exceeds 500 km by a large margin. Therefore, a d^{RANGE} of 500 km is not sufficient to establish a large number of ISLs, and, consequently, the connectivity metric does not improve by a significant amount.

Therefore, the framework prefers to maintain a relatively consistent number of CubeSats in the constellation, at the cost of modest gains in connectivity going from 4 active ISLs at 500 km to 51 at 1500 km. However, beyond this point, there is a massive increase in connectivity with the number of ISLs at 2000 km and 2500 km increasing to 1210 and 5264 respectively, offsetting the increase in the number of CubeSats, i.e., the $G_1(\cdot)$ term in the objective. As expected, a further increase in d^{RANGE} only results in a modest increase in the number of active ISLs, and once again, the system works to minimize the number of CubeSats to 476 at 3000 km. Furthermore, from Table 7.2, we note that the orbital altitude of the constellation changes in accordance with the constellation density, increasing to counter the decrease in the number of CubeSats and vice versa. On the other hand, if we set $w_1 = 0$, the framework will no longer optimize for constellation density, and the number of CubeSats in the constellation will always tend towards n_t^{MAX}, in order to maximize the coverage and connectivity metrics.

Next, as shown in Figure 7.8b, we compare the IoST constellation for $\delta = 15^\circ$ and $d^{RANGE} = 2500$ km, with other state-of-the-art constellations. Our choice of elevation angle and maximum feasible link distance for the IoST constellation follows from the fact that these metrics result in the most dense configuration. The comparison is based on the constellation quality metric, η, introduced in Section 7.4 which takes into account constellation density, coverage, and connectivity. In the interest of fairness, the normalizing parameters are given by $n_t^{MAX} = 1584$ and $n_{ISL} = 40$ based on the Starlink constellation, whose parameters reflect the maximum possible values. Furthermore, all weights w_1 through

w_4 are set to 0.25 in order to accord each metric equal priority. At the outset, IoST offers the best performance with $\eta = 1.973$, with Starlink taking second place with $\eta = 1.925$, followed by Kepler with $\eta = 1.86$. We also note that while IoST achieves a marginally higher metric than Starlink, it does so while requiring only $1/3$ of the satellites.

Furthermore, we take a look at the level of coverage and connectivity attained by each of the three constellations. Figure 7.8c, which shows the percentage coverage, brings forth a key advantage of the IoST constellation, which achieves significantly better performance than both Kepler and Starlink. In fact, IoST offers a marked 17% improvement in coverage over Kepler. In order to futher demonstrate the efficacy of the design framework, we also compare the classical coverage metric, the average revisit time, across all three constellations in Figure 7.8d. From the figure, we note that the designed IoST constellation offers the best revisit time performance across all latitudes with a revisit time of 0 sec.

A revisit time of 0 sec implies that the designed constellation is able to provide continuous gap-free coverage across all latitudes. On the other hand, Starlink's coverage is absent in the higher latitudes exceeding $67°$, while Kepler only offers continuous coverage in the polar regions. In comparing the connectivity performance, we obtain the ISL density for each constellation as shown in Figure 7.8e. The ISL density in this case has been calculated for $\beta = 1$. In this regard we note that all three constellations achieve a similar level of performance.

Thus, through a mix of new as well as classical metrics, we have demonstrated that the constellation designed by our framework achieves excellent performance in terms of both coverage and connectivity, while minimizing the CubeSats in the constellation. More generally, the results serve to ratify the optimality of the solutions obtained. Having showcased the efficacy of our proposed framework, we now leverage it for designing constellations for specific coverage use cases, as detailed next.

Table 7.3: IoST constellation design configurations for latitude-specific coverage

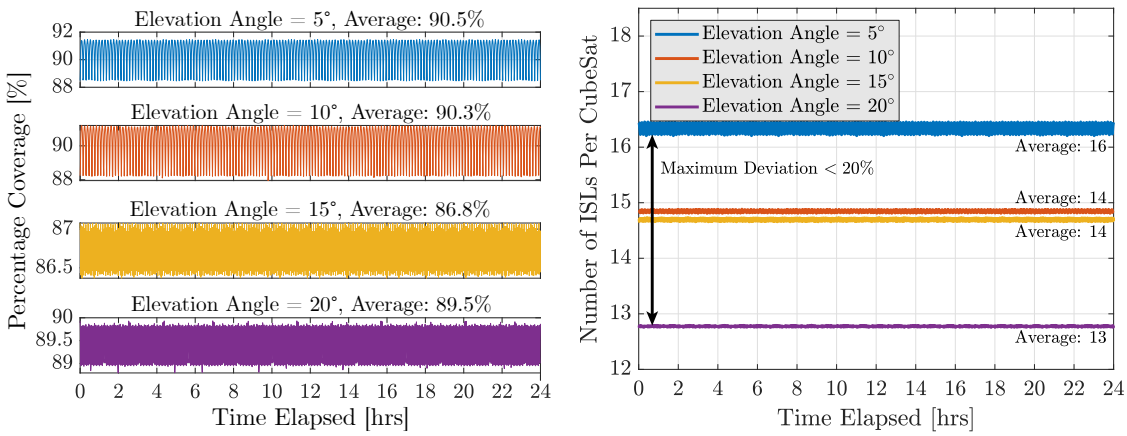
Elevation Angle Constraint (δ) $[^\circ]$	**Optimal IoST Constellation** ($\boldsymbol{x^*} = [h^*\ i^*\ n_s^*\ n_p^*\ f_p^*]^T$)
5	$[784.46\ 25.81\ 22\ 22\ 1]^T$
10	$[824.68\ 30.08\ 22\ 22\ 1]^T$
15	$[861.74\ 32.88\ 27\ 18\ 1]^T$
20	$[880.09\ 35.48\ 23\ 20\ 1]^T$

7.5.2 The Latitude-Specific Coverage Use Case

For the latitude-specific coverage use case, we leverage the proposed framework to design constellations to provide optimized coverage and connectivity within a specific set of latitudes. The results presented in this section are based on $\phi_{UB} = 50^\circ$ and $\phi_{LB} = -50^\circ$, with d^{RANGE} fixed at 2500 km. Furthermore, we vary the minimum elevation angle constraint, δ, in steps of 5° from 5° to 20°. The optimal IoST constellations thus obtained have been showcased in Table 7.3.

Unlike the global coverage use case, in the absence of comparable constellations targeting latitude-specific coverage, our primary intention in this section is to showcase the adaptability of the presented framework, i.e., as the static parameters change, the constellation design adapts to maintain a similar level of performance in terms of coverage and connectivity over the latitudes of interest. Our assertion is reinforced through the results shown in Table 7.3 and Figure 7.9.

First, from Table 7.3, we note that as the minimum elevation angle constraint increases, the orbital inclination also increases, going from 25.81° at 5° elevation to 35.48° at 20° elevation. This result can be explained by the fact that as the minimum elevation angle constraint increases, latitudes that are farther away from equator, i.e. $\phi = 0^\circ$, are no longer within the field of view of the CubeSats in the constellation. As a result, the system compensates for this change by increasing the orbital inclination, allowing for the higher latitudes to return



(a) Percentage coverage comparison for varying elevation angles.

(b) Constellation connectivity comparison for varying elevation angles.

Figure 7.9: The latitude-specific coverage use case.

to the field of view, which is necessary for maintaining adequate coverage. Furthermore, we note that the number of CubeSats within the constellation stays relatively constant at or around 470, demonstrating that a change in the elevation angle requirements brought forth by system upgrades such as a change in the on-board sensing devices, does not necessitate the deployment of additional CubeSats, in order to maintain optimality, thus saving costs.

Furthermore, Figure 7.9a shows the level of coverage achieved by the different constellations over a 24 h observation period. In line with our claim of optimizing for a similar level of performance across constellations with different elevation angle constraints, we note that the percentage coverage remains consistent across the board, in going from 90.5% at $5°$ elevation to 89.5% at $20°$ elevation. In order to further quantify the coverage performance, we also measure the average revisit time achieved by each constellation using STK. As we have noted in the previous section, here too, the average revisit time across all latitudes of interest is 0 sec, thus verifying coverage optimality.

Finally, we have the constellation connectivity characterization in Figure 7.9b. More specifically, Figure 7.9b represents the change in the number of ISLs per CubeSat over the 24 h observation period. From the figure, we see that the number of ISLs per CubeSat decreases from an average of 16 to an average of 13 as the elevation angle constraint is

Table 7.4: IoST constellation design configurations for regional coverage

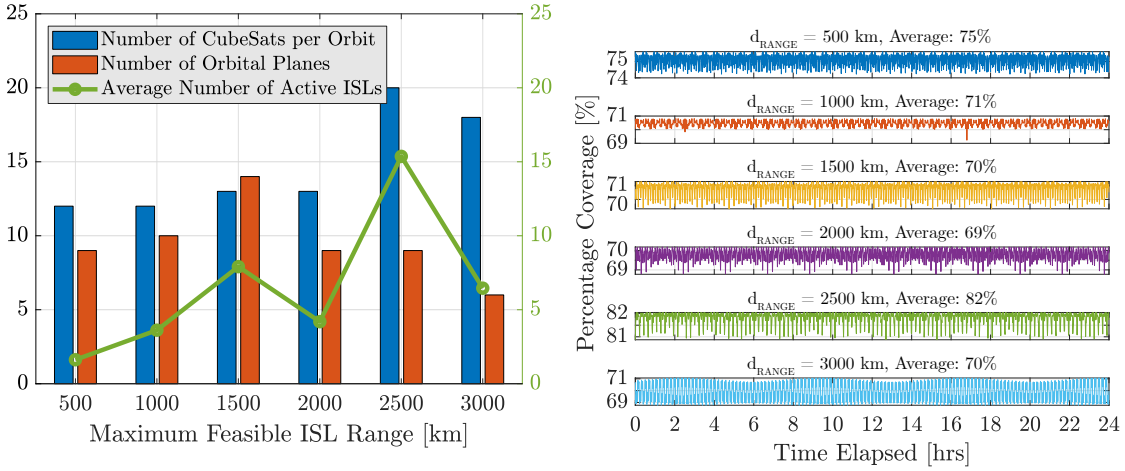
Maximum ISL Range (d^{RANGE})	**Optimal IoST Constellation** ($\boldsymbol{x^*} = [h^*\ i^*\ n_s^*\ n_p^*\ f_p^*]^T$)
500 km	$[734.52\ 35.58\ 12\ 9\ 4]^T$
1000 km	$[671.29\ 33.77\ 12\ 10\ 5]^T$
1500 km	$[622.10\ 38.02\ 13\ 14\ 10]^T$
2000 km	$[668.25\ 34.17\ 13\ 9\ 8]^T$
2500 km	$[651.09\ 37.50\ 20\ 9\ 1]^T$
3000 km	$[691.40\ 37.36\ 18\ 6\ 4]^T$

increased from 5° to 20°. This decrease can be attributed to an increase in the orbital altitude as the elevation angle constraint increases. With an increase in orbital altitude, the CubeSats grow farther apart, reducing the number of ISLs per CubeSat. However, the decrease of three ISLs on an average still represents a less than 20% deviation, indicating the similar connectivity performance of the four constellations under consideration. In this manner, with the help of the results showcased in Figure 7.9, we have demonstrated the adaptability of the framework.

7.5.3 The Regional Coverage Use Case

For the regional coverage use case, we take into consideration the bounding box that includes the continental United States. The bounds for this region are given by $\lambda_{UB} = -66.93^\circ$, $\lambda_{LB} = -125^\circ$, $\phi_{UB} = 49.59$ and $\phi_{LB} = 24.94^\circ$. As in the global coverage use case, we vary d^{RANGE} from 500 km to 3000 km for an elevation angle constraint of $\delta = 15^\circ$, with the resulting optimal constellation configurations being presented in Table 7.4. Furthermore, Figure 7.10a shows the change in the number of orbital planes, the number of CubeSats per orbit, and the number of ISLs in the constellation with change in the maximum feasible ISL range.

The figure presents a familiar trend. Relaxation in the ISL range constraint allows for an



(a) Variation in orbital inclination with change in elevation angle constraint.

(b) Percentage coverage comparison.

Figure 7.10: Region-specific coverage.

increase in the number of active ISLs and a subsequent increase in the number of CubeSats in the constellation, going from 72 at $d^{RANGE} = 500$ km to 180 at $d^{RANGE} = 2500$ km, beyond which a further relaxation of the constraint serves no advantage and the constellation size decreases. From an operational standpoint, we would prefer to have a larger number of active ISLs and therefore $d^{RANGE} = 2500$ km would represent the optimal operating point. This is further reinforced by the results in Figure 7.10b, wherein the corresponding constellation achieves 82% coverage on an average. At the same time, we note that the next best coverage is offered for $d^{RANGE} = 500$ km, despite having the least number of satellites. This result can be attributed to the orbital altitude, i.e., $h^* = 734.52$ km, of the resulting constellation, which allows for a larger spherical radius and consequently better coverage.

To summarize, by evaluating the performance of our constellation design framework for different use cases, and by benchmarking the results against the existing state-of-the-art solutions, we have demonstrated the versatility and scalability of the presented design solution. At the same time, we note that the constellations our design has been compared against are real world systems with existing physical deployments.

7.6 Highlights

In this chapter we have presented a modular and highly customizable large-scale constellation design framework for the Internet of Space Things. In doing so, we have developed robust orbit propagation, coverage estimation, and connectivity estimation modules, along with a custom LsCD annealing function. The proposed framework seeks to optimize constellation design based on CubeSat density, as well as a detailed mathematical characterization of coverage and connectivity parameters. Furthermore, we have evaluated the performance of the system for global, latitude-specific, as well as regional operational considerations. Based on the results obtained, we have demonstrated that our constellation design procedure can scale well for several hundred CubeSats and can adapt seamlessly to serve different use cases. We have also verified the efficacy of the designed IoST constellations through an extensive set of comparisons with the existing state-of-the-art. To this end, we anticipate that the work presented herein will serve as a benchmark for the design of mega-constellations.

CHAPTER 8
ONLINE SEGMENT ROUTING FOR SOFTWARE-DEFINED CUBESAT NETWORKS

In the previous chapter, we have introduced a large-scale constellation design framework for realizing the Internet of Space Things. Now that the we have this framework, we can design ultra-dense network topologies with optimized coverage and connectivity features. However, with an ultra-dense network, a key challenge is routing the resulting large volume of data. Therefore, in this chapter, we introduce a novel software-defined networking based segment routing framework for CubeSats with a view to maximizing demand satisfaction and reducing the overall network control traffic.

8.1 Motivation and Related Work

Over the past few years, CubeSats have witnessed increasing traction in the domains of aerospace systems, and communications and networking. The rising popularity of these nanosatellites has led to the possibility of large-scale CubeSat constellations serving a wide variety of applications ranging from monitoring and reconnaissance to in-space backhauling, thus necessitating an equally robust data routing framework.

At the outset, we observe that there is limited prior art concerning routing algorithms that are geared towards CubeSat networks [191]. Of particular note are the frameworks presented in [192, 193]. While [192] presents a time-expanded graph-based scheme for networks that operate based on the bent-pipe paradigm, [193] presents a variant of the contact graph routing approach for use with nanosatellites. On the other hand, there exists a rich body of literature catering to traditional LEO satellites [194]. However, a majority of the schemes presented in [194] require distributed on-board path computation, precluding their use on resource-limited CubeSats.

Instead, we have previously proposed the use of software-defined networking (SDN) for CubeSats [7]. At the same time, we recognize that, by virtue of being a long fat network (LFN), software-defined CubeSat networks are adversely affected by the proliferation of control traffic produced by signaling messages such as `PacketIn` and `FlowMod` that are exchanged between the controllers on the ground and the CubeSats in space. Therefore, any proposed SDN-based routing framework must address this issue.

Within the more general domain of SDN-based LEO satellite networks, [50] introduces the multipath TCP transport protocol, along with an accompanying routing framework, while [52] applies the developed framework for service delivery to remote naval vessels. However, the framework presented in [50] and [52] requires the use of GEO satellites, precluding its use in a CubeSat-only network. In a similar vein, the routing algorithm presented in [195] also relies on the presence of GEO satellites, thereby affecting latency performance. On the other hand, in [196], Zhu et al. introduce a software-defined routing algorithm for use with LEO satellites that seeks to minimize overall network congestion. However, the presented work does not provide solutions for reducing control traffic.

Furthermore, [197] presents an SDN-based routing solution that leverages ant colony optimization over a set of discretized time intervals. Nonetheless, the path computation algorithm presented in [197] relies on prior knowledge of the traffic matrix during each discretized time slice, rendering it sub-optimal for scenarios wherein flow requests arrive in an online manner. One of the more comprehensive works in this domain, SERvICE [48], presents a routing framework based on Delay Tolerant Networking and OpenFlow. However, SERvICE uses LEO satellites as access nodes only, instead relying on MEO satellites for data routing.

To summarize, we note the following drawbacks in the prior art: (i) emphasis on distributed on-board path computation, (ii) reliance on GEO and MEO satellites, and (iii) lack of consideration for control traffic volumes. To this end, with a view to addressing these shortcomings, we present an online intra-domain segment routing framework for software-

defined CubeSat networks in this chapter. In particular, we make use of segment routing (SR) because it has been designed to significantly reduce control traffic [155], and thus lends itself well to LFNs. More specifically, we provide the following contributions: (i) a comprehensive mathematical model of the online intra-domain segment routing problem, (ii) a robust algorithm for solving the route optimization problem, (iii) an analytical bound on the performance of the algorithm, and (iv) extensive performance evaluation metrics to demonstrate the suitability of the proposed framework.

To the best of our knowledge, this work is the first to present an SDN-based approach to routing in CubeSat networks. The remainder of this chapter is organized as follows. We present the system model in Section 8.2, followed by the problem formulation and associated online optimization framework in Section 8.3, and the results and analyses in Section 8.4. Finally, we conclude the chapter in Section 8.5.

8.2 System Model

In this section we present details regarding the construction of an equivalent network topology from a given CubeSat constellation, along with the SR model used in the proposed framework. In particular, concerning the former, we leverage the design and analysis tools developed by us in Chapter 7, to obtain key metrics associated with the network topology, such as orbital positions and link metrics. Furthermore, we exclusively focus on data routing within the domain of a single controller, i.e., state consistency considerations associated with multi-controller systems are beyond the scope of this study.

8.2.1 Topology Construction

The orbital motion of CubeSats causes rapid changes in network topology, with the resulting topological dynamism serving as a major impediment to efficient protocol design. To address this issue, the virtual topology (VT) and virtual node (VN) methods that serve to abstract these frequent topological changes have been developed. At the outset, the VT

approach is not amenable to SDN-based LFNs because of the amount of control traffic it generates. Instead, we leverage the VN approach that divides the Earth's surface into cells called VNs, and assigns a CubeSat for each VN, which is responsible for forwarding data associated with that VN.

As the CubeSat moves in its orbit, it may leave its initial VN and move into another cell, while a second CubeSat moves into the VN initially assigned to the first CubeSat. In this manner, the VNs remain static, while the actual CubeSat that is associated with a particular VN keeps changing. Path computation is then performed based on these static ground-reference VNs, as opposed to the CubeSats themselves. The primary advantage of this approach is that the controller needs to perform path computation only once per flow, and then deliver the updates to the corresponding CubeSats that move in a deterministic manner across VNs. Packet loss that may occur during VN transitions can be handled through retransmissions at the transport layer, or via a dedicated handover scheme.

Within the context of the proposed framework, we generate VNs through Voronoi tessellations that use the sub-satellite points (SSPs) of the CubeSats as generators at the time of initialization, t_i, as shown in Figure 8.1. The actual forwarding action at any given point in time, $t + t_i$, for a particular VN, is done by the CubeSat whose SSP lies closest to the generator point for that VN. Through the rest of this chapter, we use the terms VNs and CubeSats interchangeably. While the VNs represent the nodes in the topology, denoted by the set V, connectivity between VNs is realized through inter-satellite links (ISLs), that represent the edges, E.

We note that since edge construction is performed at the time of initialization, the ISLs may experience path stretch and path contraction as time passes, in addition to outages. While the former is not a major concern [198], the latter can adversely impact system performance. To this end, with a view to maintaining low complexity, we use mean orthodromic distances between CubeSats as a measure of link length, along with the probability of ISL availability as a measure of link reliability. Within this context, a link is considered absent

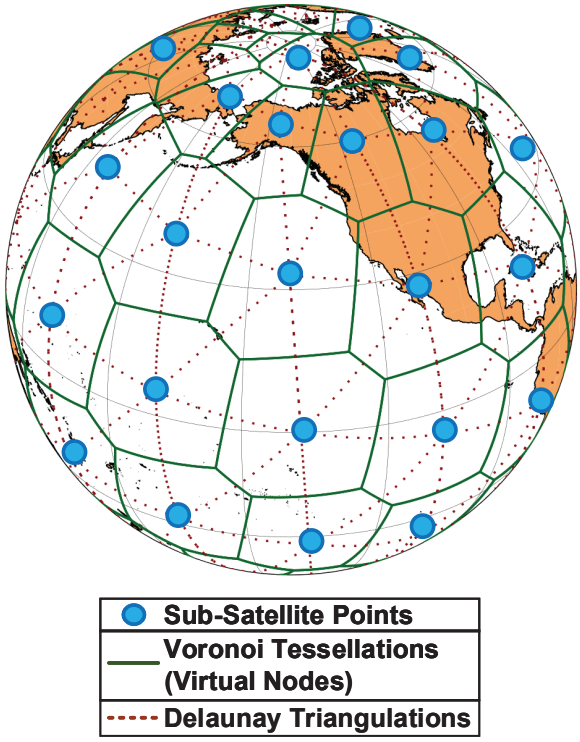


Figure 8.1: Voronoi tessellation-based virtual nodes.

if its instantaneous link length exceeds a predefined maximum feasible ISL distance, d_{ISL}. With a view to maintaining high link availability, we only select those ISLs for which the link reliability metric is above a predefined threshold r_{TH}.

The mean link distance and reliability computations are done on a per-orbital period basis at 0.01 s intervals, with the topology being reinitialized at the start of each orbital period. In this manner, we are also able to account for orbital effects such as nodal precession. Furthermore, we note that as a consequence of orbital motion, ISLs are intermittent as opposed to persistent. Therefore, we introduce a link persistence metric, $p_l \; \forall \; l \in E$, which denotes the duration of the maximum continuous link outage over a single orbital period. As described in Section 8.3, the normalized link persistence metric, $\overline{p}_l = \frac{p_l}{\max_{l \in E}\{p_l\}}$ serves as a vital component of the link weight.

In general, regarding edge construction we consider the following: (i) any ISL is considered valid so long as it does not exceed d_{ISL}, and has a link reliability metric above the threshold, r_{TH}, (ii) inter-plane ISLs are switched off in the polar regions, i.e., the lati-

tudes ranging from $\phi = \pm 70^{\circ}$ to $\phi = \pm 90^{\circ}$, and (iii) ISLs cannot be established between counter-rotating CubeSats. Thus, through the VN and edge construction procedures, we have transformed the CubeSat constellation into an equivalent undirected graph of the form $G = (V, E)$, where V is the set of VNs, and E is the set of edges.

8.2.2 Segment Routing Model

We have previously described that SR is a perfect fit for software-defined CubeSat networks due to its emphasis on the minimization of control traffic. More specifically, we utilize the middlepoint routing (MR) model in this framework, wherein routing is done based on a sequence of logical segments formed by middlepoints between the ingress and egress nodes. We note the following key terms from the MR model that are applicable to our framework:

- **Middlepoints:** Middlepoints are those VNs through which a flow must necessarily pass. A flow may pass through one or more middlepoints on its way from the ingress to the egress node.
- **Segments:** Segments are logical connections between: (i) two middlepoints, (ii) the ingress node and a middlepoint, or (iii) a middlepoint and the egress node. Each segment is realized as a single physical path that traverses a number of links between its two endpoints.
- **Tunnels:** A tunnel can be visualized a sequence of segments that begins at the ingress, traverses a number of middlepoints, before ending at the egress node.

The ground controller determines the optimal tunnel or set of optimal tunnels for every flow within its domain. Information regarding the selected tunnel is included as part of the SR header, in the form of a stack of middlepoint labels. As the flow moves from one middlepoint to another, the top label is popped off, and the flow is routed to the middlepoint indicated by the next label, as shown in Figure 8.2. The bottom-most label in the stack identifies the egress node.

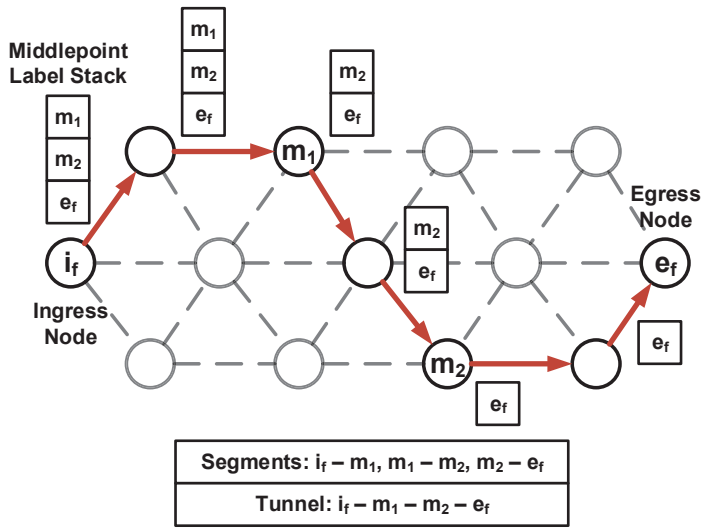


Figure 8.2: Middlepoint-based segment routing model.

Thus, every VN along the tunnel need only store flow entries pertaining to the corresponding middlepoints and egress nodes. Since each flow is no longer identified by a unique source-destination pair, but instead by a collection of middlepoints, the intermediate nodes do not need to store flow entries for each demand, thereby achieving significant reduction in control traffic. In Section 8.4, we will compare the volume of control traffic between the proposed framework and classical SDN to further elucidate the difference. Finally, it is also possible for the flow to be split across multiple tunnels between the ingress and egress nodes, thereby providing implicit support for multi-path routing and resiliency to CubeSat failure.

8.3 Problem Formulation

In this section, we define the optimal intra-domain segment routing problem for CubeSat networks, along with our approach to the solution.

8.3.1 Problem Definition

Having obtained the undirected graph $G = (V, E)$ previously, we denote the set of middlepoints by M, where $|M| \leq |V|$, i.e., every VN can serve as a candidate middlepoint.

Furthermore, every flow request $f \in F$ is characterized by an ingress node, i_f, an egress node, e_f, and a demand, d_f, which represents the bandwidth required by the flow. Furthermore, M_f denotes the number of middlepoints used for flow f, with T_f representing the set of all possible tunnels with M_f middlepoints, such that $M_f \leq |M|$. Every tunnel $t \in T_f$ is characterized by a set of segments, S_t, wherein each segment s is realized through a number of ISLs, $l \in E$. Each link, l, has a unit link-flow cost, w_l, and a link capacity, c_l. Since the intra-domain segment routing problem aims to determine the optimal set of tunnels to be used for a given flow request, we introduce the decision variable, $x_{ft} \; \forall \; f \in F, t \in T_f$, which characterizes the fraction of flow demand carried by tunnel t for flow f, such that,

$$\sum_{t \in T_f} x_{ft} = 1 \qquad \forall \; f \in F, \tag{8.1}$$

and $x_{ft} \geq 0$.

We also introduce an indicator variable, $\delta_{sl} \; \forall \; s \in S_t, l \in E$, of the form,

$$\delta_{sl} = \begin{cases} 1, \text{ if segment } s \text{ uses link } l; \\ 0, \text{ otherwise,} \end{cases} \tag{8.2}$$

which characterizes the relationship between segments and links. Furthermore, the link capacity constraint is given as,

$$g_l = \sum_{f \in F} \sum_{t \in T_f} \sum_{s \in S_t} \delta_{sl} x_{ft} d_f \leq c_l \qquad \forall \; l \in E. \tag{8.3}$$

We note that the problem definition does not take CubeSat processing capacity constraints into consideration. The reason for this is two-fold. First, since the system leverages SDN, the processing overhead associated with on-board route computation is avoided, leaving flow table lookup as the only significant computational activity that is performed on the CubeSats. Second, we note that the incoming traffic at any given CubeSat cannot exceed the aggregate capacity of the ISLs associated with that CubeSat, making it trivial to provision

appropriate processing capacity at all CubeSats during the network dimensioning stage.

In order to quantify the cost of routing over link l, we take into consideration both the normalized link persistence metric, $\overline{p}_l$, as well as the normalized link distance metric, $\overline{o_l} = \frac{o_l}{d_{ISL}}$, where o_l is the mean orthodromic link length. Accordingly, the unit link flow cost is given by, $w_l = w_1\overline{o_l} + w_2\overline{p}_l$, where $w_1 + w_2 = 1$, and $0 \leq w_1, w_2 \leq 1$. Herein, the distance metric is representative of the transmission power required to transmit over said link, while the link persistence metric represents the data storage capacity required at the corresponding source CubeSat. This follows from the fact that links with a lower persistence metric will experience outages that last longer, thus requiring greater buffer capacity at the associated CubeSats. With this, the overall link cost, y_l, is expressed as,

$$y_l = \sum_{f \in F} \sum_{t \in T_f} \sum_{s \in S_t} \delta_{sl} x_{ft} w_l \frac{d_f}{c_l} \qquad \forall\, l \in E. \tag{8.4}$$

From (8.4) we note that the link cost is jointly characterized by the link length, persistence, and capacity. Together, these three parameters serve as the joint routing metric. The corresponding utility function is then given by,

$$U_{SR} = \max_{l \in E} \{y_l :\ \forall\, l \in E\}, \tag{8.5}$$

with the associated optimization problem being given by

$$\begin{aligned} \textbf{Find:} \quad & x_{ft} \quad \forall\, f \in F,\ t \in T_f \\ \textbf{Minimize:} \quad & U_{SR} \\ \textbf{Subject To:} \quad & (8.1) \text{ and } (8.3) \end{aligned} \quad . \tag{8.6}$$

The problem presented above seeks to assign a set of tunnels to each flow request, with a view to minimize the maximum link cost, with the objective function serving two purposes: (i) load balancing, and (ii) cost minimization. The latter is necessitated by the fact that: (i) CubeSats have a limited power budget, and transmitting over unnecessarily long

distances leads to energy wastage, and (ii) CubeSats have limited data storage capabilities, and therefore links with low persistence are best avoided.

8.3.2 Pre-Processing Procedures

At the outset, from (8.6) we note that the size of the problem is significantly impacted by $|M|$, $|T_f|$, and M_f. For example, with $M_f = |M| = |V|$, the number of candidate tunnels per flow request, f, would be $\sum_{i=1}^{i=|V|} \binom{|V|}{i} i!$, i.e., for a network of 500 CubeSats, that amounts to an exponentially large number of decision variables per flow. Therefore, before attempting to develop an algorithm, it is necessary to undertake certain pre-processing procedures to make the problem more tractable.

First, in order to select the elements of set M, we leverage the concept of betweenness centrality [199]. In general, the betweeness centrality, for any VN $v \in V$, is given by

$$C_B(v) = \sum_{\substack{s \in V \\ s \neq v}} \sum_{\substack{t \in V \\ t \neq s \neq v}} \frac{\sigma_{st}(v)}{\sigma_{st}} \qquad \forall\, v \in V, \tag{8.7}$$

where $\sigma_{st}(v)$ is the number of shortest paths from s and t that pass through v and σ_{st} is the total number of shortest paths from s to t. Within the context of this chapter, path length is measured with respect to the unit link-flow cost, w_l. Furthermore, it has been demonstrated that segment routing operates well even with a relatively few number of middlepoints [155], therefore, after computing the betweeness centrality for each VN, we pick the top N_M VNs as middlepoints, such that $|M| = N_M$.

Next, we quantify the number of middlepoints per flow, M_f, using a simple distance-dependent metric of the form

$$M_f = \min \left\{ \max \left\{ \left\lfloor \frac{distance(i_f, e_f)}{d_{ISL}} \right\rfloor, 1 \right\}, |M| \right\} \;\forall\, f \in F. \tag{8.8}$$

The motivation being that, since the link length cannot exceed d_{ISL}, M_f provides a measure

of the minimum number of VNs that a flow must be routed through, fitting well with our definition of middlepoints. Furthermore, we note that since the presented framework deals exclusively with intra-domain routing, the number of middlepoints as obtained from (8.8), and consequently, the number of candidate tunnels, $|T_f|$ is not very large. Additionally, for any segment, s, between two middlepoints, the corresponding physical links that comprise this segment can be determined by calculating the shortest path, with respect to w_l, between the two middlepoints, thus determining δ_{sl}. Finally, we note that the pre-processing procedures described herein can be performed efficiently in an offline manner, before running the route optimization framework described next.

8.3.3 Online Optimization Framework

The optimization problem presented in (8.6) can be trivially linearized, and solved using a linear programming solver such as CPLEX. However, an offline solution of this kind requires a traffic matrix that describes the flow sources, destinations, and demands, in advance. In a practical scenario, however, flow requests arrive sequentially over time, and the system does not have information concerning the parameters associated with future requests, thus, necessitating an online optimization framework. The design of the proposed framework draws upon the approach outlined in [200].

We begin by restating the constraints in (8.1) and (8.3) for a scenario where the first $i-1$ requests have been routed, as follows,

$$\sum_{t \in T_f} x_{ft} = 1 \qquad \forall\, f \in \{1,\ 2,\ \ldots,\ i-1\}, \tag{8.9}$$

with $x_{ft} \geq 0$, and

$$g_l(i-1) = \sum_{f=1}^{i-1} \sum_{t \in T_f} \sum_{s \in S_t} \delta_{sl} x_{ft} d_f \leq c_l \qquad \forall\, l \in E. \tag{8.10}$$

Furthermore, the link cost including the $(i-1)$th request is given by,

$$y_l(i-1) = \sum_{f=1}^{i-1} \sum_{t \in T_f} \sum_{s \in S_t} \delta_{sl} x_{ft} w_l \frac{d_f}{c_l} \qquad \forall\, l \in E. \tag{8.11}$$

For routing the ith request through tunnel $t \in T_i$, we introduce a modified utility function of the form,

$$U_{OL}(i,t) = \sum_{s \in S_t} \sum_{l \in E} \delta_{sl} \left(\kappa^{\overline{y}_l(i-1) + \frac{w_l d_i}{c_l \Lambda}} - \kappa^{\overline{y}_l(i-1)} \right), \tag{8.12}$$

where $\kappa > 1$ and, $\overline{y}_l(\cdot) = \frac{y_l(\cdot)}{\Lambda}$, with Λ being an estimate of the optimal cost defined in (8.6). More specifically, (8.12) represents the change in the sum exponential costs given the previous $i-1$ requests, as a result of routing the ith request through tunnel t. Minimizing $U_{OL}(i)$ works well in practice because: (i) it ensures that no single link exhibits too high a cost either as a result of link weight or traffic volume, and (ii) the use of exponential costs prevents resource wastage, ensuring that a larger number of requests can be served.

Taking into consideration (8.9), (8.10), and (8.12), Algorithm 6 outlines the tunnel selection subroutine for the ith request. More specifically, the algorithm takes as input, an estimate of the optimal cost, Λ, the performance guarantee, β, the graph, G, the flow request, i, the existing costs, $y_l(i-1)$, the candidate tunnels, T_i, along with the segments and links comprising each tunnel, and attempts to find the set of minimum cost tunnels, X_i, with respect to (8.12). The validity of each tunnel is checked by verifying whether: (i) $y_l(i) \leq \beta\Lambda$, and (ii) $\sum_{f=1}^{i} \sum_{t \in T_f} \sum_{s \in S_t} \delta_{sl} x_{ft} d_f \leq c_l$, for all links that are a part of the tunnel. If a minimum cost tunnel fails to meet either criteria, it is discarded, and if all tunnels fail to meet either criteria, the request i is rejected. Concerning the performance of the algorithm, we introduce Theorem 1. Within the context of the presented theorem, λ^* represents the optimal offline cost defined in (8.6).

Theorem 1. *If there exists a $\lambda^* :\ \lambda^* \leq \Lambda$, then Algorithm 6 will always find a set of tunnels to route the incoming flow request through, with a corresponding performance guarantee*

Algorithm 6 Tunnel Selection Subroutine

Input: Λ, β, G, i_i, e_i, d_i, $y_l(i-1)$, T_i
Output: X_i
$X_i \leftarrow \emptyset$
for $t \leftarrow 1$ to $|T_i|$ **do**
  if $U_{OL}(i,t) = \min_{t \in T_i}\{U_{OL}(i,t)\}$ **then**
    $X_i \leftarrow X_i \cup \{t\}$
  end if
end for
$t \leftarrow 0$
while $t < |X_i|$ **do**
  if $\exists\, l \in t:\ y_l(i-1) + \frac{w_l d_i}{c_l |X_i|} > \beta\Lambda$ **or** $g_l(i-1) + \frac{d_i}{|X_i|} > c_l$ **then**
    $X_i \leftarrow X_i \setminus \{t\}$
    if $X_i = \emptyset$ **then**
      break
    else
      $t \leftarrow 0$
      continue
    end if
  end if
  $t \leftarrow t + 1$
end while
if $X_i = \emptyset$ **then**
  return FAIL
else
  return X_i
end if

β, *i.e.,* $y_l(i) \leq \beta\Lambda\ \forall\ l \in E$, *provided none of the selected tunnels violate the link capacity constraint.*

Proof. We begin by introducing the set X_i^* which denotes the optimal offline tunnel set for the ith flow request, along with $\overline{y}_l^*(i) = \frac{y_l^*(i)}{\Lambda}$, which denotes the normalized cost after routing i requests under the optimal offline solution. Furthermore, we define the incremental cost associated with routing the ith request over link l as $z_l(i) = \frac{w_l d_i}{c_l}$, with $\overline{z}_l(i) = \frac{z_l(i)}{\Lambda}$. Then, based on [200], we introduce the following potential function,

$$\Phi(i) = \sum_{l \in E} \kappa^{\overline{y}_l(i)} \left(\gamma - \overline{y}_l^*(i)\right), \tag{8.13}$$

where $\gamma > 1$. For the ith request, the change in the potential function is given by,

$$\begin{aligned}\Phi(i) - \Phi(i-1) &= \sum_{l \in E} \kappa^{\overline{y}_l(i)} \left(\gamma - (\overline{y}_l^*(i-1) + \overline{z}_l(i))\right) - \sum_{l \in E} \kappa^{\overline{y}_l(i-1)} \left(\gamma - \overline{y}_l^*(i-1)\right) \\ &= \sum_{l \in X_i} \left(\kappa^{\overline{y}_l(i)} - \kappa^{\overline{y}_l(i-1)}\right) \left(\gamma - \overline{y}_l^*(i-1)\right) - \sum_{l \in X_i^*} \kappa^{\overline{y}_l(i)} \overline{z}_l(i).\end{aligned} \tag{8.14}$$

Since, $\overline{y}_l^*(i-1) \geq 0$, from (8.14), we have

$$\begin{aligned}\Phi(i) - \Phi(i-1) &\leq \sum_{l \in X_i} \gamma \left(\kappa^{\overline{y}_l(i-1) + \overline{z}_l(i)} - \kappa^{\overline{y}_l(i-1)}\right) - \sum_{l \in X_i^*} \kappa^{\overline{y}_l(i)} \overline{z}_l(i) \\ &\leq \sum_{l \in X_i^*} \gamma \left(\kappa^{\overline{y}_l(i-1) + \overline{z}_l(i)} - \kappa^{\overline{y}_l(i-1)}\right) - \sum_{l \in X_i^*} \kappa^{\overline{y}_l(i)} \overline{z}_l(i) \\ &= \sum_{l \in X_i^*} \kappa^{\overline{y}_l(i-1)} \left[\gamma \left(\kappa^{\overline{z}_l(i)} - 1\right) - \overline{z}_l(i)\right],\end{aligned} \tag{8.15}$$

where the second inequality follows from the fact that X_i represents the optimal set with respect to $\kappa^{\overline{y}_l(i-1) + \overline{z}_l(i)} - \kappa^{\overline{y}_l(i-1)}$, and replacing X_i with X_i^* leads to a higher overall cost.

Next, we show that the potential function does not increase with each incoming request, i.e., $\Phi(i) - \Phi(i-1) \leq 0$. Since, $\forall\ l \in X_i^*$, $0 \leq \overline{z}_l(i) \leq \frac{\lambda^*}{\Lambda} \leq 1$, therefore, we need only show that $\gamma \left(\kappa^{\overline{z}_l(i)} - 1\right) - \overline{z}_l(i) \leq 0$. This condition holds true when $\kappa \leq 1 + \frac{1}{\gamma}$. Furthermore, with $\overline{y}_l^*(i) \leq \frac{\lambda^*}{\Lambda} \leq 1$, $\kappa > 1$, and $\Phi(i) \leq \Phi(i-1)$, after routing the ith request, we have

$$(\gamma - 1)\kappa^{\max_{l \in E} \overline{y}_l(i)} \leq (\gamma - 1) \sum_{l \in E} \kappa^{\overline{y}_l(i)} \leq \Phi(i) \leq \Phi(0), \tag{8.16}$$

where, $\Phi(0) \leq \gamma |E|$. The result in (8.16) can be restated as $\max_{l \in E} y_l(i) \leq \beta \Lambda$, where

$$\beta = \log_\kappa \left(\frac{\gamma |E|}{\gamma - 1}\right). \tag{8.17}$$

□

Thus, we can leverage Algorithm 6 for our framework. The fundamental idea behind

our framework can be summarized as follows. If Algorithm 6 returns FAIL, flow request i is rejected, and the estimated optimal cost, Λ, is doubled with all existing costs being set to 0, i.e., $y_l(i) \leftarrow 0$, in line with the doubling approach [200]. The performance of the framework is given in terms of its competitive ratio in Proposition 1. The competitive ratio of an online algorithm is the worst-case ratio between the cost of the solution found by the algorithm to the cost of the optimal offline solution, which considers all flow requests at once, based on the availability of a traffic matrix.

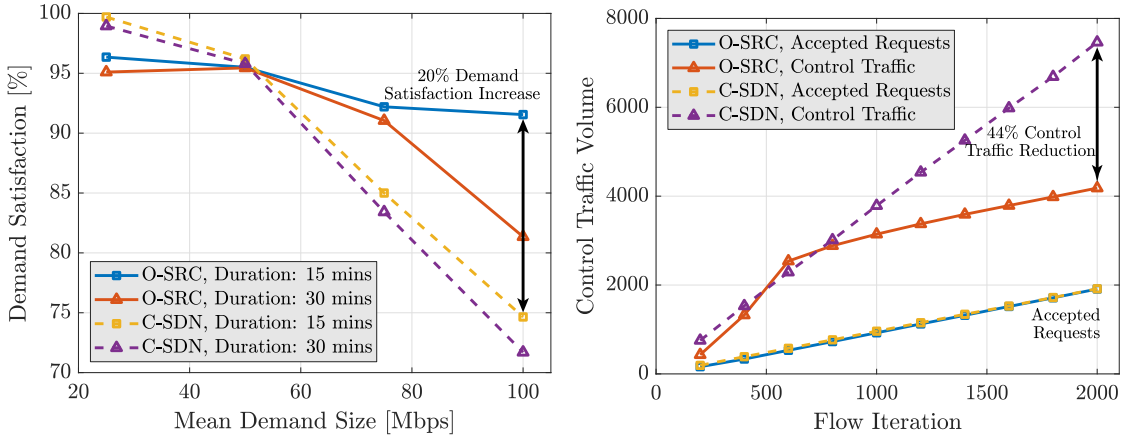
Proposition 1. *The competitive ratio of the proposed online optimization framework is of the form* $\mathcal{O}(\log |E|)$.

Proof. On account of the doubling approach, we have $\Lambda \leq 4\lambda^*$. Combining this result with Theorem 1, we have $\max_{l \in E} y_l(i) \leq 4\beta\lambda^*$. Thus, the competitive ratio is of the form $\mathcal{O}(\log |E|)$. □

8.4 Results and Analyses

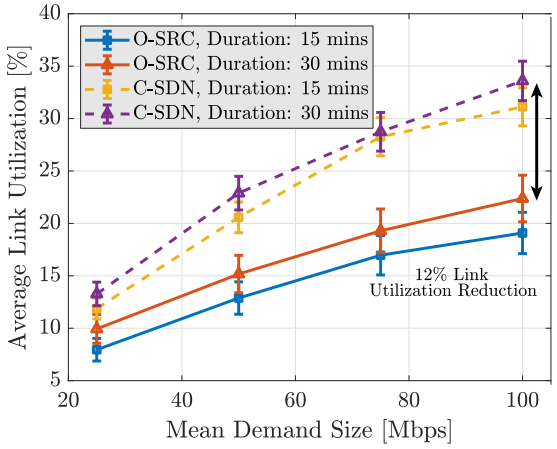
In this section, we present a comprehensive performance evaluation of the proposed online segment routing framework, and benchmark the results obtained against a classical SDN framework that relies on shortest path route computation popular in existing literature. More specifically, we base our performance evaluation on the following metrics: (i) demand satisfaction, (ii) control traffic volume, and (iii) average link utilization. All simulation results are obtained using the Starlink constellation that consists of 1584 satellites at an altitude of 550 km with an inclination of $53°$.

We generate 2000 flows across 100 trials from a Poisson distribution with a mean rate of 1 arrival per minute, and random source destination pairings within the continental United States. We plot the average values for all three aforementioned metrics in Figure 8.3, using the acronym O-SRC to denote our framework, and C-SDN to denote classical SDN. The flow demands and duration values are generated from a uniform distribution and an



(a) Demand satisfaction.

(b) Control traffic volume.



(c) Average link utilization.

Figure 8.3: Performance evaluation metrics.

exponential distribution respectively, while the ISL capacities (in Mbps) are obtained from a uniform distribution of the form $U(50, 1000)$. The other parameters are set as follows: $d_{ISL} = 850$ km, $r_{TH} = 0.7$, $|N_M| = 8$, and $\gamma = 2$. First, in Figure 8.3a, we plot the variation in demand satisfaction as the mean flow demand is varied from 25 Mbps to 100 Mbps, and the mean flow duration is varied from 15 minutes to 30 minutes. The demand satisfaction value represents the percentage of flows for which a suitable route can be found, i.e., the percentage of accepted requests. From the figure, we note that the demand satisfaction decreases as the mean demand and mean duration increase. This result follows from the fact that as flows consume more bandwidth and last longer, the network becomes saturated,

increasing the flow rejection rate. However, even in the worst case scenario O-SRC is able to accept 20% more flows than the C-SDN system, highlighting the robustness of our solution.

Next, in Figure 8.3b, we compare the control traffic volumes across O-SRC and C-SDN, during the time elapsed between the first and the last flow requests. The control traffic volume represents the number of messages that are exchanged between the controller and the CubeSats in order to successfully route the incoming flows. The comparison in Figure 8.3a is based on a mean demand of $50\ \mathrm{Mbps}$ and a mean duration of $30\ \mathrm{minutes}$ due to a similar level of request acceptance across segment routing and classical SDN. From the figure, it is apparent that O-SRC achieves a significant reduction in control traffic, with a control traffic volume that is nearly half that of C-SDN.

Figure 8.3c presents the average link utilization with a 95% confidence interval for varying flow demands and durations. As expected, an increase in either the demand or the duration leads to an increase in the link utilization. However, O-SRC consistently achieves a lower link utilization than C-SDN, highlighting the load balancing capabilities of our framework. In particular, we note that for a mean demand of $100\ \mathrm{Mbps}$ and a mean duration of $15\ \mathrm{minutes}$, O-SRC achieves a 12% lower link utilization, while achieving a 20% higher demand satisfaction than C-SDN, i.e., our framework is able to route a larger number of flows while keeping the average link utilization low. Thus, we have demonstrated the suitability of the proposed framework for resource constrained LFNs.

8.5 Highlights

In this chapter we have introduced an SDN based intra-domain SR framework for CubeSats. As part of this framework, we have provided a robust analytical characterization of the underlying mathematical problem, and presented an online near-optimal route computation algorithm. We have also derived a competitive ratio for the proposed online algorithm, and benchmarked the performance of our framework against classical SDN systems. Based on the results obtained, we conclude that not only does our framework achieve a higher level

of demand satisfaction but also provides a significant reduction in control traffic, along with load balancing.

CHAPTER 9
AUTOMATIC NETWORK SLICING FOR SPACE-GROUND INTEGRATED NETWORKS

Having introduce a large-scale topology design solution and a robust segment routing framework in the previous chapters, we now turn to the deployment of applications and services over the IoST infrastructure in this chapter. This chapter of the thesis presents a novel automatic network slicing framework that has been specifically designed with space-ground integrated networks in mind. At the outset, we note that some of the concepts presented in this chapter build upon the ideas introduced in Chapter 8, however, the content has been reproduced here for sake of completeness.

9.1 Motivation

Satellite communications centered around CubeSats has been recognized as a key enabling technology for the upcoming sixth generation of wireless systems [1]. In order to leverage the potentially ubiquitous connectivity offered by CubeSats, in Chapter 6, we have introduced a cyber-physical system known as the Internet of Space Things (IoST) that consists of an ultra-dense network of CubeSats augmented with a robust ground network. As discussed previously, the use cases of IoST can be classified into three categories based on functionality as follows:

- **Monitoring and Reconnaissance:** CubeSats equipped with a variety of imaging sensors can be used for monitoring and reconnaisance within IoST. Terrain and asset monitoring is a key application enabled by this use case. Reconnaissance capabilities are also vital for disaster monitoring in regions with unstable geology wherein the monitoring of buildings, roads, and bridges is of utmost importance.

- **In-space Backhaul:** An in-space backhaul is of great importance in remote areas that usually lack terrestrial communications infrastructure. Furthermore, IoST can also prove to be particularly useful in ensuring the continuity of critical communications in the case of emergencies such as tornadoes or earthquakes wherein the ground infrastructure might be subject to damage.

- **Cyber-physical Integration:** IoST seeks to achieve cyber-physical integration through a combination of data collected from both local and remote sensors, which is then fed to data analytics frameworks in the cyber space for additional insights. Remote industrial automation serves as a good example for this use case. Within this context, IoST can leverage aerial reconnaissance for monitoring industrial sites, remote connectivity for automating factory operations, and localized sensors for equipment monitoring. This data can then be delivered to a centralized operations center for real-time processing.

To this end, the aforementioned use cases necessitate a network architecture that can serve a wide variety of application scenarios with differing service-level agreement (SLA) requirements over the same physical infrastructure in an end-to-end manner. At the same time, since the different use cases might belong to a variety of different stakeholders, IoST must support multi-tenancy and functional isolation of services. Consequently, a network slicing framework is vital to the success of IoST.

Building upon our work in Chapter 8, for the first time in the literature, this chapter introduces a novel automatic network slicing framework for space-ground integrated networks (SGINs) of which IoST is a prime example. In particular, the "automatic" keyword here implies that our proposed network slicing framework is purely SLA-based and does not require prior information regarding the resource requirements associated with a slice as is often the case with several state-of-the-art slicing frameworks. An automatic network slicing framework offers a number of advantages. At the outset, tenants no longer need to model their slices in terms of explicit resource requirements. Instead, within our framework a slice is described in terms of its SLA, which is a more quantifiable metric. The importance

of transitioning from a resource-based model to an SLA-based system is also underscored by the fact that the same slice can have differing resource requirements based on the specifications of the underlying infrastructure, and therefore, an automatic framework such as the one proposed herein can help abstract the complexities of deploying slices over distributed non-homogeneous hardware. Furthermore, the use of an SLA-based framework allows for slice customization, i.e., slices can be customized to address different use cases, and slice priorities can be adjusted based on operational considerations.

More generally, the objective of this framework is to find the optimal IoST Gateways and CubeSats required per slice, along with the corresponding allocation of resources such that the SLA violation associated with a given slice is minimized. The realization of a framework of this kind requires several interrelated challenges to be solved. Before deploying network slices onto the underlying infrastructure, we first need to convert the CubeSat constellation into an equivalent network topology. Towards this objective, we introduce a novel Voronoi tessellation-based topology construction mechanism to map the CubeSat-based constellation to a corresponding network topology. In this manner, the resulting topology serves as the underlying substrate over which the different slices are deployed.

The next challenge relates to developing an analytical characterization of the proposed slicing framework. Here, we formulate the automatic network slicing problem as a mixed integer nonlinear program (MINLP). Then, recognizing the complexity associated with an MINLP, we split the original problem into two subproblems relating to route computation and resource allocation respectively. On the one hand, the route computation subproblem deals with determining the specific gateway nodes and CubeSats over which a given slice must be deployed. On the other hand, the resource allocation subproblem deals with determining the optimal allocation of resources relating to the nodes and links over which a given slice is deployed.

In order to solve the route allocation subproblem, we propose a novel segment routing-based online algorithm which also serves as an admission control mechanism. Then, once

we have a set of admitted slices along with their corresponding routes, we finally solve a linearized version of the resource allocation problem in the interest of computational efficiency. The proposed automatic slicing framework is also evaluated in the context of a variety of application scenarios relating to the aforementioned use cases, with a view to benchmark system performance in terms of resource utilization, SLA metrics, resource distribution, and service priority impacts.

To summarize, the major contributions of this chapter are as follows:

- A novel topology construction mechanism based on Voronoi tessellations to map ultra-dense constellations to equivalent network topologies.
- A comprehensive analytical formulation of the automatic network slicing problem within the context of SGINs addressing the dual objectives of route computation and resource allocation with minimal SLA violations.
- A robust and flexible SLA model to characterize the nuances associated with slices relating to a wide variety of use cases.
- A novel online segment routing-based approach to route computation and admission control characterized by minimal signaling overhead.
- A comprehensive performance evaluation of the proposed slicing framework in the context of typical application scenarios associated with IoST.

The remainder of this chapter is organized as follows. Section 9.2 surveys the state of the art concerning software-defined networking and network slicing solutions with an emphasis on SGINs. Section 9.3 presents the SGIN system model under consideration along with the preliminaries relating to topology construction. Section 9.4 introduces the automatic network slicing framework along with the solutions for routing and resource allocation. The evaluation scenario featuring slices corresponding to different use cases

is presented in Section 9.5, along with the associated analyses and discussion. Finally, Section 9.6 concludes the chapter.

9.2 Related Work

At the outset, IoST is part of a larger effort to push the boundaries of space systems in general, and SGINs in particular. We note the presence of several complementary works in the domain of satellite-focused software-defined networking (SAT-SDN) systems, in addition to efforts targeting optical and extremely high frequency (EHF) connectivity for satellites. Within the SAT-SDN realm, Ferrus et al. [44] provide a detailed description of the use cases and benefits associated with SDN and NFV, along with the architectural frameworks proposed in [45] and [46]. In addition, the SERvICE framework [47, 49] introduces a novel integrated space-terrestrial satellite network that leverages LEO, MEO, and GEO satellites. The past few years have also witnessed advancements in optical connectivity for satellites [201, 202, 203, 204, 205]. In particular, concerning the space segment, Chen et al. [202] and Chaudhary et al. [203] have demonstrated the feasibility of multi-gigabit optical links, while system optimization concerning the ground network has been addressed at great length in [204] and [205]. Furthermore, recent research efforts have also focused on extending terahertz connectivity to small satellites with novel transceiver designs [143] and resource allocation strategies [206].

More recently, network slicing for SGINs has gained traction with Drif et al. presenting a detailed slicing framework for satellite integration within 5G [207]. In particular, the work presented in [207] focuses on system design aspects along with management and orchestration solutions for integrating satellite networks within 5G. The authors in the aforementioned chapter have considered an application scenario involving the use of a satellite-based transport network between terrestrial radio access and core networks. Within this context, they present primitives for slicing the space segment through the introduction of a satellite management system and a slice classifier.

On the analytical front, [208] introduces an MILP-based model that aims to minimize the amount of infrastructure resource that is assigned to network slices subject to certain resource availability and latency constraints. The system model presented in [208] considers a single satellite between two terrestrial endpoints. In this case, network function deployment is limited to terrestrial nodes, with the satellite under consideration providing a transport link. In a similar vein, the authors in [209] and [210] present a network slicing solution focused on eMBB services within SGINs, along with machine learning algorithms for resource allocation.

While existing solutions are largely focused on systems with a single satellite, the framework proposed in this chapter is centered around an ultra-dense network of CubeSats with a special emphasis on use case driven network slice customization augmented with a robust SLA model to achieve our goal of automatic network slicing.

9.3 System Model and Preliminaries

9.3.1 Overview

A space-ground integrated network (SGIN) typically consists of both terrestrial and non-terrestrial segments. Within the context of IoST, the terrestrial network infrastructure consists of IoST Hubs and Gateways along with the Customer Premises, while the non-terrestrial infrastructure consists of CubeSats. The network slicing framework proposed in this chapter is deployed at the IoST Hubs which house a majority of the network control infrastructure, and, consequently, we use the terms IoST Hubs and controllers interchangeably throughout this chapter. Depending on the use case, the network slices that are deployed on this infrastructure can have endpoints that either in space or on the ground.

For example, a monitoring and reconnaissance use case has a non-terrestrial source coupled with a terrestrial destination. On the other hand, for the in-space backhauling use case, both endpoints are terrestrial with the non-terrestrial network linking the two endpoints. Additionally, we note that the framework presented in this chapter caters exclusively to

slices having both endpoints within the domain of the same IoST Hub. Before beginning our discussion about the slicing framework, we note that while the topology of the terrestrial network is fixed, the orbital motion of CubeSats causes rapid changes in the non-terrestrial topology. The topological dynamism that arises from these frequent changes is a major impediment to efficient protocol design. To address this issue, we introduce the topology construction mechanism described in the next section.

9.3.2 Topology Construction

At the outset, the general idea of any topology construction mechanism for satellite systems is to abstract the temporal changes that occur within the network. To this end, the virtual topology (VT) and virtual node (VN) methods have proved popular. The VT method considers a set of discrete time intervals, with a fixed topology associated with each interval. However, the VT approach is not amenable to SDN-based CubeSat networks because the controller must perform slice route computation for each time interval under consideration, and then deliver the flow table updates to the corresponding CubeSats. For a fairly large network consisting of several hundred CubeSats [9], the resulting control traffic and corresponding TCAM requirements render this approach unsuitable.

On the other hand, the VN approach divides the Earth's surface into cells called virtual nodes, and assigns a CubeSat for each VN, which is responsible for processing and forwarding data associated with that VN. As a given CubeSat moves in its orbit, it may leave its initial VN and move into another cell, while a second CubeSat moves into the VN initially assigned to the first CubeSat. In this manner, while the actual CubeSat that is associated with a particular VN keeps changing, the VNs themselves remain static. The deployment of slices is thus performed based on these static ground-reference VNs, as opposed to the CubeSats themselves.

The aforementioned VN approach offers the significant advantage that the controller needs to perform path computation and resource allocation only once per slice, and then

deliver the updates to the corresponding CubeSats that move in a deterministic manner across VNs. The fact that CubeSats are built to a standard specification ensures that all CubeSats are identical in terms of hardware, thus guaranteeing that the corresponding VNs do not differ from each other in their resource capacities. Data loss that may occur during VN transitions can be handled through retransmissions at the transport layer, or via a dedicated slice migration and handover scheme, which falls outside the scope of this work.

Algorithm 7 Voronoi Tessellation Based Virtual Node Generation and Mapping Framework

for $i \leftarrow 1$ to n_t **do**
  $(\boldsymbol{\lambda_s},\ \boldsymbol{\phi_s}) \leftarrow \text{PROPAGATE}(\boldsymbol{f_s}, T)$
end for
for $t \leftarrow t_i$ to $t_i + |T| - 1$ **do**
  if $t = t_i$ **then**
    $\mathfrak{VD}(t) \leftarrow \text{VORONOI}(\boldsymbol{\lambda}(t),\ \boldsymbol{\phi}(t))$
  end if
  for $v \leftarrow 1$ to $|\mathcal{V}_{\mathcal{NT}}|$ **do**
    $\mathfrak{m}_v(t) \leftarrow \arg\min_{s \in \{1,\ ...,\ n_t\}} \frac{\sum_{j=1}^{|V(\mathfrak{VD}_v(t))|} dist(\boldsymbol{j}, \mathfrak{p}_\mathfrak{s}(\mathbf{t}))}{|V(\mathfrak{VD}_v(t))|}$
  end for
end for

The primarily challenge associated with the VN approach lies in the fact that it requires a robust VN generation and mapping scheme. To this end, we introduce a novel Voronoi tessellation-based node generation and mapping framework as outlined in Algorithm 7. First, Algorithm 7 uses the orbit propagation subroutine developed by us previously [9] to obtain the subsatellite points (SSPs) associated with all CubeSats in the constellation for the entire orbital period denoted by the discretized set T, wherein $\boldsymbol{f_s}$ represents the orbital configuration of CubeSat s. In general, the set of SSPs for CubeSat s is denoted by $\mathfrak{p}_\mathfrak{s} = (\boldsymbol{\lambda_s}, \boldsymbol{\phi_s})$. Furthermore, $\boldsymbol{\lambda_s} \coloneqq \{\lambda_s(t)\}_{t=1}^{T}$ and $\boldsymbol{\phi_s} \coloneqq \{\phi_s(t)\}_{t=1}^{T}$ represent the longitudes and latitudes associated with the SSP of CubeSat $s \in \{1,\ 2,\ \ldots,\ n_t\}$, where n_t is the total number of CubeSats in the constellation. Then, starting with the time of initialization t_i, the algorithm generates a Voronoi diagram $\mathfrak{VD}(t)$ using the SSPs at $t = t_i$ as generators, where $\boldsymbol{\lambda(t)} \coloneqq \{\lambda_s(t)\}_{s=1}^{n_t}$ and $\boldsymbol{\phi(t)} \coloneqq \{\phi_s(t)\}_{s=1}^{n_t}$. The resulting diagram has been shown in Figure 9.1. The individual Voronoi regions, $\mathfrak{VD}_\mathfrak{v}(t)$, that constitute

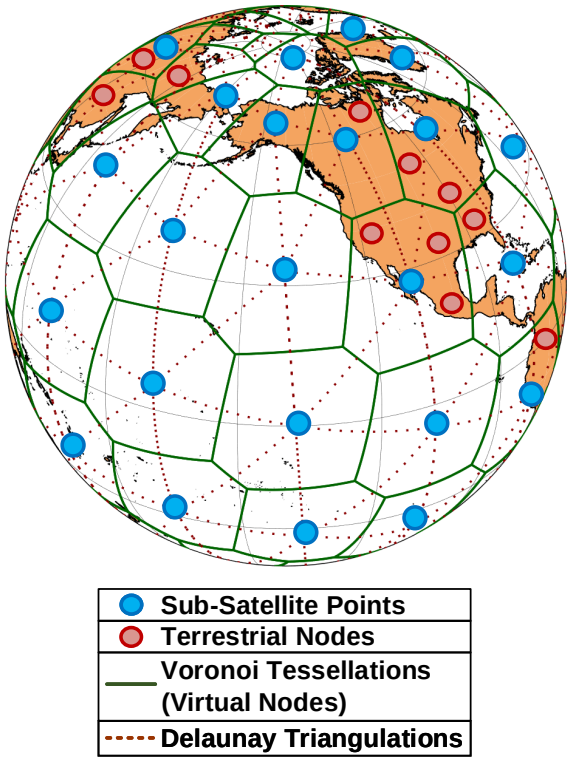


Figure 9.1: Voronoi tessellation-based virtual nodes.

$\mathfrak{VD}(t)$ represent the virtual nodes denoted by $v \in \mathcal{V}_{\mathcal{NT}}$.

Once the VNs have been generated, at each time step t, the algorithm maps CubeSats to the corresponding VNs by determining the average orthodromic distance from the vertices of a given Voronoi region, $V(\mathfrak{VD}_v(t))$, to the SSPs of the CubeSats in the constellation. The CubeSat s offering the minimum such distance is then mapped to VN v and is thus responsible for the slices deployed at v, with the mapping being denoted by $\mathfrak{m}_v(t)$, i.e.,

$$\mathfrak{m}_v(t) = \underset{s \in \{1,\ ...,\ n_t\}}{\arg\min} \frac{\sum_{j=1}^{|V(\mathfrak{VD}_v(t))|} dist(\boldsymbol{j}, \mathfrak{p}_{\mathfrak{s}}(\mathbf{t}))}{|V(\mathfrak{VD}_v(t))|} \quad \forall\ v \in \mathcal{V}_{\mathcal{NT}}. \tag{9.1}$$

In this manner, the VNs represent the nodes in the non-terrestrial topology, denoted by the set $\mathcal{V}_{\mathcal{NT}}$. On the other hand, connectivity between VNs is realized through ISLs that represent the edges $e \in \mathcal{E}_{\mathcal{NT}}$.

Since edge construction is performed at $t = t_i$, we note that ISLs may experience path stretch or path contraction over time, in addition to outages. At the same time, we note that

catering to real-time ISL lengths and availability will result in significant overhead [211]. While it has been shown that path stretch and path contraction are not major concerns [198], ISL outages can significantly impact system performance for the worse. To overcome this challenge, we use mean orthodromic distances between CubeSats as measure of link length, along with the probability of ISL availability as a measure of link reliability. An approach of this kind helps maintain low system complexity while accurately characterizing ISL behavior.

Under this approach, an ISL is marked absent if its instantaneous length exceeds a predefined value, d_{ISL} or if the two CubeSats in question are not within line-of-sight (LoS). We note that the value of d_{ISL} is a function of the link budget, and thus depends upon a variety of factors ranging from the transmit power and frequency bands in use, to the channel model and the antenna design of the CubeSat. However, instead of specifying the link budget, we prefer to use the generic parameter d_{ISL}, with the idea that a system designer can determine the radio communication range specific to their scenario and simply substitute the appropriate value for d_{ISL}. In order to quantify visibility for any two CubeSats, s and k, with $s \neq k$, we note that if their respective position vectors, $\boldsymbol{r_s(t)}$ and $\boldsymbol{r_k(t)}$, form an acute angle, then the CubeSats are within are LoS. However, if this angle is obtuse, as shown in Figure 9.2, then the associated link tangential height $h_{sk}(t)$ must be greater than the radius of the Earth, R_E, to ensure visibility.

Furthermore, we only select those ISLs for which the link reliability metric is above a predefined threshold r_{TH}. This ensures that ISL outages are minimized in the resulting topology. The link availability and reliability are computed on a per-orbital period basis at $1/|T|$ intervals and the topology is reinitialized at the start of each orbital period. This approach works well in practice because the operational duration of a given slice is typically lower than the orbital period, which, for CubeSats deployed even as low as $500\ \mathrm{km}$, is well over an hour. By operating on a per-orbital period basis, we are also able to account for orbital effects such as nodal precession. Finally, we note that since ISLs are intermittent as

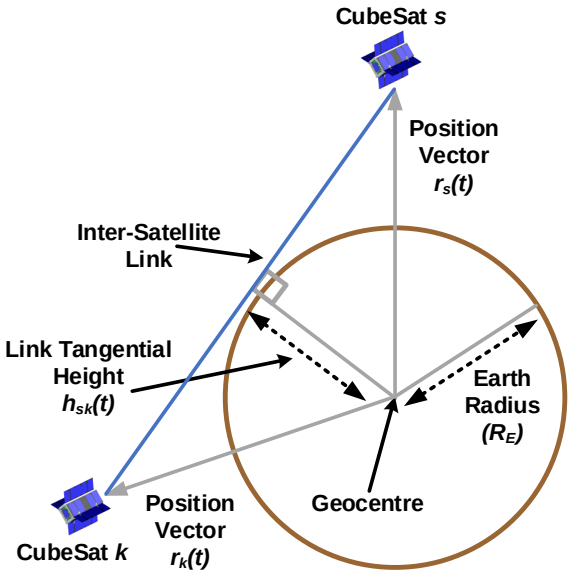


Figure 9.2: CubeSat visibility condition for ISL availability.

opposed to persistent, we introduce a link persistence metric, $p_e \; \forall \; e \in \mathcal{E}_{\mathcal{NT}}$, which denotes the duration of the maximum continuous link outage over a single orbital period. As we will see in Section 9.4, a normalized version of this metric, $\overline{p}_e = \frac{p_e}{\max_{e \in \mathcal{E}_{\mathcal{NT}}}\{p_e\}}$, is used as a key component of the link weight.

Leveraging the aforementioned metrics, the non-terrestrial edge construction procedure can be summarized as:

- An ISL is considered valid so long as its length does not exceed d_{ISL}, it has a link reliability metric above r_{TH}, and its constituent CubeSats are within LoS.
- ISLs across orbital planes are switched off in the polar region, i.e., the latitudes ranging from $\phi = 70°$ to $\phi = 90°$.
- ISLs cannot be established between counter-rotating CubeSats.

In this manner, through the topology construction procedure described in this section, a CubeSat constellation can be efficiently converted to an equivalent non-terrestrial topology with set of nodes $\mathcal{V}_{\mathcal{NT}}$ and edges $\mathcal{E}_{\mathcal{NT}}$. Then, the overall topology of the SGIN can be obtained as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} \triangleq \mathcal{V}_{\mathcal{T}} \cup \mathcal{V}_{\mathcal{NT}}$ and $\mathcal{E} \triangleq \mathcal{E}_{\mathcal{T}} \cup \mathcal{E}_{\mathcal{NT}}$. Here, $\mathcal{V}_{\mathcal{T}}$ and $\mathcal{E}_{\mathcal{T}}$

represent the terrestrial infrastructure. Furthermore, $\mathcal{E}_{\mathcal{NT}}$ is also expanded to include GSLs that are established between terrestrial nodes and their corresponding VNs.

9.3.3 System Model

As part of defining the system model, we note that in addition to the network infrastructure denoted by sets $\mathcal{V}$ and $\mathcal{E}$, the set of slices $\mathcal{S}$ is also a key component of the framework. The computing capacity at a given node $v \in \mathcal{V}$ is denoted by μ_v. In particular, computing capabilities are provided by the gateways in case of terrestrial nodes and by the Command and Data Handling (C&DH) subsystem in case of non-terrestrial nodes. Within the context of this chapter, computing capacity is defined as the number of requests that a node is able to serve per unit time, where a single packet is considered equivalent to a request. Additionally, the number of computing cycles per kilobit at node v is given by τ_v. The node resource model also includes the power budget κ_v. We assume that while CubeSats are power limited, terrestrial nodes do not exhibit power limitations, i.e., $\kappa_v \rightarrow \infty \; \forall \; v \in \mathcal{V}_{\mathcal{T}}$.

On the other hand, the key resource for a given link e relates to link capacity, which is represented by c_e. For terrestrial links, the link capacity is part of the link definition, however, non-terrestrial links are defined in terms of their bandwidth b_e, with capacity being obtained as $c_e = b_e \log_2 (1 + \gamma)$, where γ is the threshold SINR. More generally,

$$c_e = \alpha_e b_e \log_2 (1 + \gamma) + (1 - \alpha_e) \, c_e \qquad \forall \; e \in \mathcal{E}, \tag{9.2}$$

where α_e is an auxiliary variable that characterizes the nature of link e, i.e., $\alpha_e = 1$, if $e \in \mathcal{E}_{\mathcal{NT}}$ and 0 otherwise.

Having defined the primary resources associated with the underlying infrastructure, we now turn our attention to the slice model. Since, the proposed framework seeks to implement automatic network slicing, a given slice s is characterized solely in terms of its SLA, in addition to the ingress and egress endpoints, v_{si} and v_{se}. First, the traffic associated with

slice s is modeled by a regenerative stochastic process with rate $\lambda_s > 0$, which describes the packet arrivals. On the other hand, the packet lengths are exponentially distributed with mean $\theta_s > 0$. The target SLA for throughput can then be obtained as $r_s^{SLA} = \lambda_s \theta_s$.

The target SLA for latency is denoted by δ_s^{SLA}, with the throughput-latency trade-off being denoted by Γ_s. More specifically, Γ_s can be used to prioritize either throughput or latency in the slice definition. For example, $\Gamma_s = 0.5$ denotes equal priority, while $\Gamma_s = 0.7$ indicates a higher priority for throughput. Furthermore, recognizing that the target SLA is not always achievable, we introduce the metrics w_{sr} and $w_{s\delta}$ which represent the throughput and latency priorities respectively, i.e., if r_s^{SLA} represents the target throughput, then $w_{sr} r_s^{SLA}$ represents the threshold throughput. Finally, the SLA metric w_s represents the overall slice priority for slice s with respect to other slices in the network.

Furthermore, we assume that the system operates in discrete time, i.e., for slices that are to be instantiated at time t, the corresponding slice characteristics are communicated to the respective IoST Hubs during the interval $(t-1,\ t)$. The proposed slicing framework operates within a specific time slice, and thus, going forward, the absence of t in the notation denotes a generic time instance.

9.4 Theoretical Framework

In this section we first formulate the automatic network slicing problem in Section 9.4.1, followed by the route computation and resource allocation procedures in Sections 9.4.2 and 9.4.3 respectively.

9.4.1 Problem Formulation

We begin the automatic network slicing problem formulation by defining indicator variables $\alpha_{sv}, \forall\ s \in \mathcal{S},\ v \in \mathcal{V}$ and $\alpha_{se}, \forall\ s \in \mathcal{S},\ e \in \mathcal{E}$ as follows

$$\alpha_{sv} = \begin{cases} 1, \text{ if slice } s \text{ is deployed on node } v; \\ 0, \text{ otherwise,} \end{cases} \tag{9.3}$$

and,

$$\alpha_{se} = \begin{cases} 1, \text{ if slice } s \text{ is deployed on link } e; \\ 0, \text{ otherwise.} \end{cases} \tag{9.4}$$

Together, α_{sv} and α_{se} represent the path computation variables since they characterize the deployment of slice s on the underlying infrastructure. The set $\mathcal{V}_{\mathcal{S}} \subseteq \mathcal{V}$ represents the set of nodes on which the slice is deployed, with $\mathcal{E}_{\mathcal{S}}$ representing the edge set.

Next, we introduce the decision variables $0 \leq x_{\mu sv} \leq 1, \forall\, s \in \mathcal{S},\ v \in \mathcal{V}$ and $0 \leq x_{cse} \leq 1, \forall\, s \in \mathcal{S},\ e \in \mathcal{E}$ which characterize the resource allocation aspects. More specifically, $x_{\mu sv}$ represents the proportion of computing resource on node v that is allocated to slice s, and x_{cse} represents the proportion of link capacity associated with link e that is allocated to slice s. Next, in order to ensure that slices are only allocated resources on those infrastructure elements that host that specific slice, we introduce the following constraints,

$$x_{\mu sv} \leq \alpha_{sv} \qquad \forall\, s \in \mathcal{S}, v \in \mathcal{V}, \tag{9.5}$$

and

$$x_{cse} \leq \alpha_{se} \qquad \forall\, s \in \mathcal{S}, e \in \mathcal{E}. \tag{9.6}$$

Having defined the key variables associated with the problem, we now turn our attention to the metrics of delay, throughput, and power consumption. First, we note that slice delays arise as a consequence of data transmission over links $e \in \mathcal{E}_{\mathcal{S}}$ and data processing at nodes $v \in \mathcal{V}_{\mathcal{S}}$. Given that we have modeled the network traffic as a regenerative arrival process and that the packet lengths are exponentially distributed, the average delay associated with link e depends upon the traffic carried by that link, in addition to its length, d_e [212, §4.3].

Consequently, the average link delay can be expressed as

$$\delta_e = \frac{1}{c_e - \sum_{s \in \mathcal{S}} \alpha_{se} c_e x_{cse}} + \frac{d_e}{c} \qquad \forall\, e \in \mathcal{E}, \tag{9.7}$$

where $\sum_{s \in \mathcal{S}} \alpha_{se} c_e x_{cse}$ represents the data traffic carried by link e and c is the speed of light. Furthermore, in order to prevent infinite delays over any given link, we introduce the following constraint,

$$\sum_{s \in \mathcal{S}} \alpha_{se} c_e x_{cse} \leq c_e - \epsilon \qquad \forall\, e \in \mathcal{E}, \tag{9.8}$$

where $\epsilon \to 0$ is a small positive constant. The constraint represented by (9.8) ensures that the slice data traffic allocated to link e is always less than the link's capacity, guaranteeing that the system remains stable.

Furthermore, concerning the processing delay encountered at an infrastructure node, we leverage the procedure outlined in [213], which assumes a linear resource-delay dependency for each slice-node combination. More formally, the processing delay experienced by slice s at node v is modeled as

$$\delta_{sv} = \frac{\delta_v^{MAX} - \delta_v^{MIN}}{\mu_{sv}^{MIN} - \mu_{sv}^{MAX}} \mu_v x_{\mu sv} + \frac{\delta_v^{MIN} \mu_{sv}^{MIN} - \delta_v^{MAX} \mu_{sv}^{MAX}}{\mu_{sv}^{MIN} - \mu_{sv}^{MAX}} \forall\, s \in \mathcal{S},\ v \in \mathcal{V}, \tag{9.9}$$

where δ_v^{MIN} is the processing delay that results when slice s is allocated the maximum possible computing resource on node v, i.e., μ_{sv}^{MAX} or higher. On the other hand, δ_v^{MAX} is the maximum possible processing delay resulting from a minimal resource allocation of μ_{sv}^{MIN} or lower. However, in order to enforce a practical operating range in terms of computing resource allocation, we introduce the following constraints,

$$\mu_v x_{\mu sv} \geq \mu_{sv}^{MIN} + \epsilon = \lambda_s + \epsilon \qquad \forall\, s \in \mathcal{S},\ v \in \mathcal{V}, \tag{9.10}$$

and,

$$\mu_v x_{\mu sv} \leq \mu_{sv}^{MAX} = \mu_v \qquad \forall\, s \in \mathcal{S},\ v \in \mathcal{V}. \tag{9.11}$$

Constraint (9.10) ensures that node v has sufficient processing capacity for slice s, while Constraint (9.11) ensures that the node is not over-subscribed. We also introdouce a constraint on the overall node computing capacity as follows,

$$\sum_{s \in \mathcal{S}} \alpha_{sv} x_{\mu sv} \leq 1 \qquad \forall\, v \in \mathcal{V}. \tag{9.12}$$

Finally, the overall delay for slice s can be expressed as

$$\delta_s = \sum_{v \in \mathcal{V}} \alpha_{sv} \delta_{sv} + \sum_{e \in \mathcal{E}} \alpha_{se} \delta_e \qquad \forall\, s \in \mathcal{S}. \tag{9.13}$$

Having characterized link capacity for both terrestrial and non-terrestrial links in (9.2), the achievable throughput, r_{se}, for slice s over a given link e can be simply expressed as $r_{se} = c_e x_{cse}\ \forall\ s \in \mathcal{S},\ e \in \mathcal{E}$. We note that the throughput is not impacted by the node computing capacity allocated to slices as a consequence of (9.10). The effective throughput for slice s can then be expressed as

$$r_s = \min_{e \in \mathcal{E}_\mathcal{S}} r_{se} \qquad \forall\, s \in \mathcal{S}. \tag{9.14}$$

Another critical resource in CubeSat networks is electrical power. At the outset, CubeSats are power-limited devices with power budgets typically not exceeding 10 W. In general, a CubeSat's power budget accounts for power consumption due to the presence of key subsystems such as communications, C&DH, Attitude Determination and Control (ADC), and payloads. However, the payload power consumption is negligible compared to that of the communications, C&DH, and ADC subsystems [214]. Consequently, the total power consumption at a non-terrestrial node $v \in \mathcal{V}_{\mathcal{NT}}$ can be expressed as $P_v = P_{tv} + P_{\mu v} + P_{cv}$, where P_{tv} represents the transmit power, $P_{\mu v}$ is the power consumed by the C&DH subsystem, and P_{cv} is the average ADCS power. More specifically, the transmit power consumption at

node v can be modeled as [215]

$$P_{tv} = \sum_{s \in \mathcal{S}} \sum_{e \in \mathcal{E}_{\mathcal{S}} : e_s = v} b_e N_{0w} \gamma x_{cse} \qquad \forall\, v \in \mathcal{V}_{\mathcal{NT}}, \tag{9.15}$$

where N_{0w} is the noise spectral density (including interference from other CubeSats) at the receiving node $w \neq v$.

More generally, (9.15) represents the transmit power at node v required to maintain a target SINR of γ for all slices deployed on that node. On the other hand, based on [216], the C&DH power consumption due to slice s can be expressed as

$$P_{\mu sv} = \phi_v \left(\theta_s \mu_v \tau_v\right)^3 x_{\mu sv}^3 \qquad \forall\, s \in \mathcal{S},\ v \in \mathcal{V}_{\mathcal{NT}}, \tag{9.16}$$

where ϕ_v is the processor-dependent effective capacitance coefficient of node v, with $P_{\mu v} = \sum_{s \in \mathcal{S}} \alpha_{sv} P_{\mu sv}\ \forall\ v \in \mathcal{V}_{\mathcal{NT}}$. Additionally, since the ADCS power consumption is independent of the data processing and communications taking place on the CubeSat, it is represented by the constant P_{cv}. Finally, in order to enforce the power budget, we introduce the following power-related constraint for $\forall\ v \in \mathcal{V}_{\mathcal{NT}}$,

$$P_v \leq \kappa_v \qquad \forall\, v \in \mathcal{V}_{\mathcal{NT}}. \tag{9.17}$$

Given the plethora of use cases that IoST is expected to serve and the use case specific SLA model introduced previously, we note that the utility functions within this framework should be representative of the same dynamism. The key idea here is that "utility" is an application-specific construct, i.e., the return associated with a cellular backhauling slice which prioritizes high throughput is very different from an emergency communications slice which prioritizes lower latency. Consequently, it is imperative that the slice utilities within our formulation reflect this distinction. To this end, we introduce throughput and latency utility functions that operate on a sliding scale as defined below.

First, the throughput utility for slice s is defined as

$$\mathcal{U}_{sr} = \frac{r_s}{r_s^{SLA} - w_{sr} r_s^{SLA}} - \frac{w_{sr}}{1 - w_{sr}} \qquad \forall\ s \in \mathcal{S}. \tag{9.18}$$

We note that the utility value increases linearly from 0 to 1 as the slice throughput r_s increases from $w_{sr} r_s^{SLA}$ to r_s^{SLA}. As described previously, $w_{sr} r_s^{SLA}$ represents the threshold throughput while r_s^{SLA} represents the target. Furthermore, if the slice throughput drops below the threshold value, then the throughput utility associated with that slice turns negative. In order to prevent $\mathcal{U}_{sr} \to \infty$, we introduce the following constraint on slice throughput,

$$r_s \leq r_s^{SLA} \qquad \forall\ s \in \mathcal{S}, \tag{9.19}$$

which ensures that $\mathcal{U}_{sr} \leq 1$. In a similar vein, the latency utility for slice s is obtained as

$$\mathcal{U}_{s\delta} = \frac{w_{s\delta} \delta_s}{w_{s\delta} \delta_s^{SLA} - \delta_s^{SLA}} - \frac{1}{w_{s\delta} - 1} \qquad \forall\ s \in \mathcal{S}, \tag{9.20}$$

i.e., as the slice latency increases, the latency utility for that slice begins to fall, reaching a value of 0 at the threshold $\delta_s = \frac{\delta_s^{SLA}}{w_{s\delta}}$. Here too, the condition $\mathcal{U}_{s\delta} \leq 1$ is enforced through a similar constraint,

$$\delta_s \geq \delta_s^{SLA} \qquad \forall\ s \in \mathcal{S}. \tag{9.21}$$

Leveraging the throughput latency trade-off, Γ_s, the overall utility for slice s can then be expressed as

$$\mathcal{U}_s = \Gamma_s \mathcal{U}_{sr} + (1 - \Gamma_s) \mathcal{U}_{s\delta} \qquad \forall\ s \in \mathcal{S}. \tag{9.22}$$

With the problem definition complete, we summarize the SGIN-Automatic Network Slicing (SGIN-ANS) problem below, along with an overview of the variables and parameters

Table 9.1: Summary of problem parameters and decision variables

Notation	Description
α_{sv}	Binary indicator variable that takes value 1 if slice s is deployed on node v
α_{se}	Binary indicator variable that takes value 1 if slice s is deployed on link e
$x_{\mu sv}$	Continuous decision variable that quantifies the proportion of computing resource on node v that is allocated to slice s
x_{cse}	Continuous decision variable that quantifies the proportion of capacity on link e that is allocated to slice s
μ_v	Computing capacity of node v
τ_v	Number of computing cycles per kilobit at node v
ϕ_v	Effective capacitance coefficient of node v
κ_v	Power budget of node v
P_v	Overall power consumption at node v
δ_{sv}	Processing delay for slice s on node v
c_e	Capacity of link e
b_e	Bandwidth of link e
δ_e	Average delay over link e
λ_s	Average traffic arrival rate for slice s
θ_s	Average packet size for slice s
r_s	Throughput of slice s
r_s^{SLA}	Target SLA for throughput of slice s
w_{sr}	Throughput priority for slice s
δ_s	Latency of slice s
δ_s^{SLA}	Target SLA for latency of slice s
$w_{s\delta}$	Latency priority for slice s
Γ_s	Throughput-latency trade-off for slice s
w_s	Overall priority of slice s
$\mathcal{U}_{sr}$	Throughput utility for slice s
$\mathcal{U}_{s\delta}$	Latency utility for slice s
$\mathcal{U}_s$	Overall utility for slice s

in Table 9.1.

SGIN-Automatic Network Slicing (SGIN-ANS)

Find: $\alpha_{sv}, x_{\mu sv}, \alpha_{se}, x_{cse} \quad \forall\ s \in \mathcal{S},\ v \in \mathcal{V},\ e \in \mathcal{E}$

Maximize: $\mathcal{U} = \sum_{s \in \mathcal{S}} w_s \mathcal{U}_s$

Subject To: (9.5), (9.6), (9.8), (9.10) − (9.12), (9.17), (9.19), and (9.21).

In particular, the SGIN-ANS problem seeks to maximize overall system utility characterized by the SLA, subject to key constraints relating to node and link assignments, i.e., (9.5) and (9.6), link capacity, i.e., (9.8), node computing capacity, i.e., (9.10)-(9.12), node power budget, i.e., (9.17), and the target throughput and latency metrics, i.e., (9.19) and (9.21). The problem presented above is modeled as a mixed integer non-linear program (MINLP) characterized by several non-linear constraints and a large number of variables. Thus, the problem as presented is intractable. To overcome this issue, we split SGIN-ANS into two subproblems.

The first subproblem, which we refer to as the route computation problem, deals with determining the values of indicator variables α_{sv} and α_{se} as described in Section 9.4.2. In solving this problem we note that it is critical that the deployed slices do not result in link oversubscription causing the system to become unstable in line with (9.7). Therefore, in determining the optimal links and nodes, the route computation procedure must ensure that no single link in the system is oversubscribed. In other words, if the deployment of a given slice s causes the system to become unstable, then that slice should be rejected. In this manner, the route computation step also serves as an admission control mechanism.

Furthermore, we note that since the requests for the slices to be deployed at time t are communicated to the IoST Hubs during the interval $(t-1,\ t)$, the route computation procedure should function in an online manner, admitting or rejecting slice requests as they arrive. Once the online route computation procedure for $(t-1,\ t)$ is complete, the values of α_{sv} and α_{se} for all admitted slices to be instantiated at time t are known. At this point, using

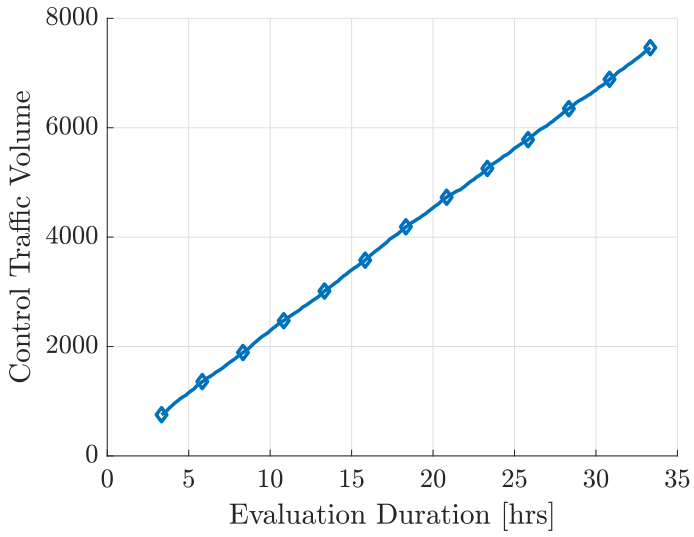


Figure 9.3: Control traffic volume for SDN system operation.

the admitted slice parameters as input, we solve the resource allocation problem which is presented as a linearized version of SGIN-ANS with α_{sv} and α_{se} as auxiliary variables as described in Section 9.4.3.

9.4.2 Online Route Computation and Admission Control

As part of designing the route computation and admission control scheme, we note that IoST is heavily centered on SDN. While SDN offers several advantages within the context of resource-limited CubeSats, by virtue of being a long fat network (LFN), software-defined CubeSat networks are adversely affected by the proliferation of control traffic produced by messages such as `PacketIn` and `FlowMod` that are exchanged between the controllers on the ground and the CubeSats in space.

In order to underscore the severity of the issue, we perform a simple experiment. Using the 1584 satellite Starlink constellation as base, we generate 2000 slice requests over a 30 hour period with randomized demands and source-destination pairings across the US. Under this scenario we track the evolution of control traffic volume in terms of the number of requests exchanged between the IoST Hubs and CubeSats over time. The results have been showcased in Figure 9.3 and validate our assertion. With a near linear increase in control

traffic over time, vanilla SDN is not suitable for CubeSat networks, and consequently SGINs, due to the resulting large volume of signaling messages.

To this end, we propose the use of SDN augmented with segment routing (SR) to overcome the control traffic problem. Our use of SR is motivated by the fact that it has been designed to significantly reduce control traffic [155], and thus lends itself well to LFNs. More specifically, within the context of this chapter, we utilize the middlepoint SR model [217], wherein routing is done based on a sequence of logical segments between the ingress and egress nodes. The key terms associated with the MR model are described next.

- **Middlepoints:** Middlepoints are those nodes through which a slice's flows must necessarily pass. A flow may pass through one or more middlepoints on its way from the ingress to the egress node.

- **Segments:** Segments are logical connections between: (i) two middlepoints, (ii) the ingress node and a middlepoint, or (iii) a middlepoint and the egress node. Each segment is realized as a single physical path that traverses a number of links between its two endpoints.

- **Tunnels:** A tunnel can be visualized a sequence of segments that begins at the ingress, traverses a number of middlepoints, before ending at the egress node.

The IoST Hub determines the most appropriate tunnel for every slice within its domain. Information regarding the selected tunnel is included as part of the SR header, in the form of a stack of middlepoint labels. As the flows associated with a slice move from one middlepoint to another, the top label is popped off, and the flows are routed to the middlepoint indicated by the next label, as explained in Chapter 8. The bottom-most label in the stack identifies the egress node.

In this manner, every node along the tunnel need only store flow table entries pertaining to the middlepoints. Since each flow is no longer identified by a unique source-destination pair associated with a slice, but instead by a collection of middlepoints, the intermediate

nodes do not need to store flow entries for each demand. Moreover, as time passes, and additional flow table entries are added, eventually a given node will come to hold information about nearly every middlepoint in the network, thereby achieving significant reduction in control traffic.

Problem Definition

In order to define the route allocation and admission control problem, we introduce certain additional parameters. First, we denote the set of middlepoints in the network by $\mathcal{M}$, where $|\mathcal{M}| \leq |\mathcal{V}|$, i.e., every node can serve as a candidate middlepoint. Furthermore, $|\mathcal{M}_s|$ denotes the number of middlepoints used for slice s, with $\mathcal{T}_s$ representing the set of all possible tunnels with $|\mathcal{M}_s|$ middlepoints, where $|\mathcal{M}_s| \leq |\mathcal{M}|$. Every tunnel $t \in \mathcal{T}_s$ is characterized by a set of segments, $\mathcal{K}_t$, wherein each segment k is realized through a number of links $e \in \mathcal{E}$.

Then, we introduce the decision variable $x_{st} \; \forall \; s \in \mathcal{S}, t \in \mathcal{T}_s$ which represents the proportion of demand associated with slice s that is carried by tunnel t, such that

$$\sum_{t \in \mathcal{T}_s} x_{st} = 1 \qquad \forall \; s \in \mathcal{S}, \tag{9.23}$$

and $x_{st} \geq 0$. While we have modeled x_{st} as a continuous variable in the interest of maintaining a generalized formulation for the route allocation problem, in practice, system designers can place an upper bound on the number of tunnels to be used for a given slice as we will see in Section 9.4.2. For example, if each slice is restricted to just one tunnel, then x_{st} takes the form of a binary indicator variable. More generally, since any given tunnel is effectively an ordered set of links and nodes, α_{sv} and α_{se} can be trivially obtained from x_{st}.

Next, we introduce an auxiliary variable, $\delta_{ke}\ \forall\ k \in \mathcal{K}_t$, $e \in \mathcal{E}$, of the form,

$$\delta_{ke} = \begin{cases} 1, \text{ if segment } k \text{ uses link } e; \\ 0, \text{ otherwise,} \end{cases} \tag{9.24}$$

which characterizes the relationship between segments and links. With the variables defined, we can express the overall link utilization within the context of the route allocation problem as follows

$$g_e = \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}_s} \sum_{k \in \mathcal{K}_t} \delta_{ke} x_{st} d_s \leq c_e \qquad \forall\ e \in \mathcal{E}, \tag{9.25}$$

where $d_s = w_{sr} r_s^{SLA}$. Then, we introduce a unit link flow cost, w_e, of the form $w_l = w_1 \overline{o_l} + w_2 \overline{p}_l$, where $w_1 + w_2 = 1$, and $0 \leq w_1, w_2 \leq 1$. Here $\overline{p}_l$ represents the normalized link persistence metric introduced in Section 9.3.2, while $\overline{o_l} = \frac{o_l}{d_{ISL}}$, where o_l is the mean orthodromic link distance. On the other hand, for terrestrial links, $w_e \to 0$ since link length and availability are not major concerns in the terrestrial scenario. Accordingly, the overall link cost is expressed as

$$y_e = \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}_s} \sum_{k \in \mathcal{K}_t} \sum_{e \in \mathcal{E}} \delta_{ke} x_{st} w_e \frac{d_s}{c_e} \qquad \forall\ e \in \mathcal{E}. \tag{9.26}$$

Thus, link cost is jointly characterized by the link length, persistence, and capacity. Together, these three parameters serve as the joint routing metric. The associated utility function can be expressed as

$$\mathcal{U}_{\text{ROUTE}} = \max_{e \in \mathcal{E}} \{ y_e :\ \forall\ e \in \mathcal{E} \}, \tag{9.27}$$

with the overall route computation and admission control problem being given by

SGIN-Route Computation and Admission Control

(SGIN-RCAC)

Find:	$x_{st} \quad \forall\, s \in \mathcal{S},\ t \in \mathcal{T}_s$
Minimize:	$\mathcal{U}_{\text{ROUTE}}$
Subject To:	(9.23) and (9.25).

The problem presented above seeks to assign a set of tunnels to each slice request, with a view to minimize the maximum link cost, with the objective function serving two purposes: (i) load balancing, and (ii) cost minimization.

Having defined the route computation problem, we note that the problem dimension is significantly impacted by $|\mathcal{M}|$, $|\mathcal{T}_s|$, and $|\mathcal{M}_s|$. If these metrics are left unbounded, the problem will end up having an exponentially large number of variables making it impossible to solve. Therefore, before attempting to solve the presented problem, we discuss some strategies relating to the selection of candidate middlepoints and the number of tunnels. First, the elements of the set $|\mathcal{M}|$ should be selected based on the use case. For example, in a single controller system, we could select the top $\mathcal{N}_M$ nodes as middlepoints, using metrics such as centrality [199] to identify the most well connected nodes. On the other hand, if we consider a nationwide use case with multiple IoST Hubs, it would make more sense to select those VNs that can geographically cover the entire region of interest, as the candidate middlepoints.

Similarly, a flow can be routed through either a single tunnel or a set of tunnels depending on the specific requirements. For example, in a network slicing scenario, we would prefer to have a single tunnel per slice in order to eschew the operational complexities associated with parallelized slice operation across multiple tunnels. In this case, $\mathcal{N}_T = 1$, where $\mathcal{N}_T$ represents the required number of tunnels in the optimal tunnel set. However, for use cases where functional isolation between services is not required and, consequently, the network

is not sliced, multiple tunnels might be the preferred option. To this end, we will examine the trade-offs between tunnel selection strategies in Section 9.4.2, providing deeper insight into the pros and cons of each.

Online Optimization Framework

The SGIN-RCAC problem can be trivially linearized, and solved using a linear programming solver such as CPLEX [218]. However, an offline approach of this kind would require knowledge of the slice endpoints and demands in advance. We have previously noted that slice requests for instance t are communicated to the IoST Hubs in an online manner during the interval $(t-1,\ t)$, i.e., upon receiving a request, we would like to immediately determine whether it can be admitted within the system and inform the slice requestor accordingly. For example, if a slice request is rejected, the requestor can adjust the slice parameters and submit the request once again during the same interval. An online admission control framework allows for this kind of flexibility. The design of this online framework draws upon the approach outlined in [200].

We begin by restating the constraints in (9.23) and (9.25) for a scenario where the first $i-1$ slice requests have been admitted, as follows,

$$\sum_{t\in\mathcal{T}_s} x_{st} = 1 \qquad \forall\ s \in \{1,\ 2,\ \ldots,\ i-1\}, \tag{9.28}$$

with $x_{st} \geq 0$, and

$$\sum_{s=1}^{i-1}\sum_{t\in\mathcal{T}_s}\sum_{k\in\mathcal{K}_t} \delta_{ke} x_{st} d_s \leq c_e \qquad \forall\ e \in \mathcal{E}. \tag{9.29}$$

Similarly, the link cost including the $(i-1)$th slice request is given by,

$$y_e(i-1) = \sum_{s=1}^{i-1}\sum_{t\in\mathcal{T}_s}\sum_{k\in\mathcal{K}_t}\sum_{e\in\mathcal{E}} \delta_{ke} x_{st} w_e \frac{d_s}{c_e} \qquad \forall\ e \in \mathcal{E}. \tag{9.30}$$

For assigning the ith request to tunnel $t \in \mathcal{T}_i$, we introduce a modified utility function

of the form,

$$U_{OL}(i,t) = \sum_{k \in \mathcal{K}_t} \sum_{e \in \mathcal{E}} \delta_{ke} \left(\kappa^{\overline{y}_e(i-1) + \frac{w_e d_i}{c_e \Lambda}} - \kappa^{\overline{y}_e(i-1)} \right), \tag{9.31}$$

where $\kappa > 1$ and $\overline{y}_e(\cdot) = \frac{y_e(\cdot)}{\Lambda}$, with Λ being an estimate of the optimal cost associated with the original SGIN-RCAC problem. In particular, (9.31) represents the change in the sum exponential costs given the previous $i-1$ requests, as a result of assigning the ith request to tunnel t. Minimizing $U_{OL}(i,t)$ works well in practice because: (i) it ensures that no single link exhibits too high a cost either as a result of link weight or traffic volume, and (ii) the use of exponential costs prevents resource wastage, ensuring that a larger number of requests can be served.

Taking into consideration (9.28), (9.29), and (9.31), Algorithm 8 outlines the tunnel selection subroutine for the ith request. More specifically, the algorithm takes as input, an estimate of the optimal cost, Λ, a performance guarantee, β, the graph, G, the slice request, i, the existing costs, $y_e(i-1)$, the candidate tunnels, T_i, the required number of tunnels in the optimal tunnel set, $\mathcal{N}_T$, along with the segments and links comprising each tunnel, and attempts to find the set of minimum cost tunnels, Y_i, with respect to (9.31). The validity of each tunnel is checked by verifying whether: (i) $y_e(i) \leq \beta\Lambda$, and (ii) $\sum_{s=1}^{i} \sum_{t \in \mathcal{T}_s} \sum_{k \in \mathcal{K}_t} \delta_{ke} x_{st} d_s \leq c_e$, for all links that are a part of that particular tunnel. If a minimum cost tunnel fails to meet either criteria, it is discarded, otherwise it is added to the optimal set X_i. However, if all tunnels fail to meet either criteria, the request i is rejected. Concerning the performance of the algorithm, we introduce Theorem 2. Within the context of the presented theorem, λ^* represents the optimal offline cost associated with the SGIN-RCAC problem.

Theorem 2. *If there exists a $\lambda^* : \lambda^* \leq \Lambda$, then Algorithm 8 will always find a set of tunnels to assign to the incoming slice request, with a corresponding performance guarantee β, i.e., $y_e(i) \leq \beta\Lambda \; \forall \; e \in \mathcal{E}$, provided none of the selected tunnels violate the link capacity constraint.*

Proof. The proof for Theorem 2 has been presented in Section 8.3.3. □

Algorithm 8 Tunnel Selection Subroutine

Input: Λ, β, G, v_{si}, v_{se}, d_s, $y_e(i-1)$, T_i, $\mathcal{N}_T$
Output: X_i
$Y_i \leftarrow \emptyset$
$X_i \leftarrow \emptyset$
if $|T_i| \leq \mathcal{N}_T$ **then**
  $\mathcal{N}_T \leftarrow |T_i|$
end if
for $t \leftarrow 1$ to $|T_i|$ **do**
  if $U_{OL}(i,t) = \min_{T_i(t) \in T_i}\{U_{OL}(i,t)\}$ **then**
    $Y_i \leftarrow Y_i \cup \{T_i(t)\}$
  end if
end for
$t \leftarrow 0$
while $t < \mathcal{N}_T$ **do**
  if $\exists\, e \in Y_i(t) :\ y_e(i-1) + \frac{w_e d_i}{c_e \mathcal{N}_T} > \beta\Lambda$ **or** $g_e(i-1) + \frac{d_i}{\mathcal{N}_T} > c_e$ **then**
    $Y_i \leftarrow Y_i \setminus \{Y_i(t)\}$
    if $Y_i = \emptyset$ **then**
      break
    else
      $t \leftarrow 0$
      continue
    end if
  end if
  for $e \in Y_i(t)$ **do**
    $y_e(i) = y_e(i-1) + \frac{w_e d_i}{c_e \mathcal{N}_T}$
    $g_e(i) = g_e(i-1) + \frac{d_i}{\mathcal{N}_T}$
  end for
  $X_i \leftarrow X_i \cup \{Y_i(t)\}$
  $t \leftarrow t + 1$
end while
if $X_i = \emptyset$ **then**
  return FAIL
else
  return X_i
end if

Thus, we can leverage Algorithm 8 for the online route allocation and admission control framework described in Algorithm 9.

On the arrival of a new slice request, Algorithm 9 calls the tunnel selection subroutine outlined in Algorithm 8 using the incoming slice request's parameters. If Algorithm 8

Algorithm 9 Online Route Allocation and Admission Control Framework

```
1: Λ ← Λ_0
2: i ← 0
3: loop
4:    if New Slice Request then
5:       Call Algorithm 8 with parameters associated with slice i
6:       if Algorithm 8 returns FAIL then
7:          Reject slice i
8:          Λ ← 2Λ
9:          for e ∈ E do
10:            y_e(i) ← 0
11:         end for
12:      else
13:         Admit slice i with tunnel set X_i
14:      end if
15:      i ← i + 1
16:   end if
17: end loop
```

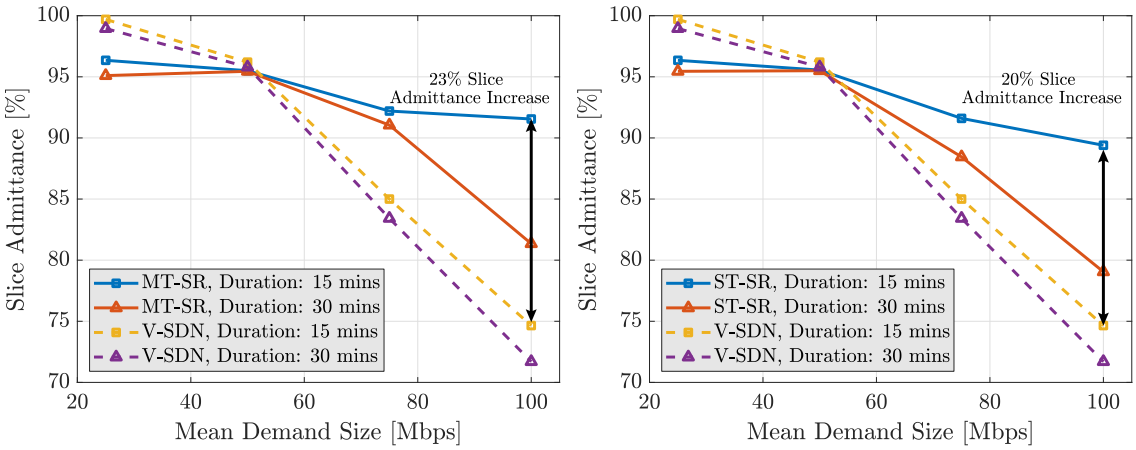
returns FAIL, slice request i is rejected, and the estimated optimal cost, Λ, is doubled with all existing costs being set to 0, i.e., $y_e(i) \leftarrow 0$, in line with the doubling approach [200]. On the other hand, if Algorithm 8 returns a set of tunnels, X_i, then the corresponding slice is accepted. The performance of the framework is given in terms of its competitive ratio in Proposition 2. The competitive ratio of an online algorithm is the worst-case ratio between the cost of the solution found by the algorithm to the cost of the optimal offline solution, which considers all slice requests at once, based on the availability of a slice request matrix.

Proposition 2. *The competitive ratio of the proposed online route allocation and admission control framework is of the form* $\mathcal{O}(\log |\mathcal{E}|)$.

Proof. The proof for Proposition 2 has been presented in Section 8.3.3. □

Validation

Having developed an online algorithm for the SGIN-RCAC problem, in this section, we validate our claims of improved network performance and reduced control traffic in comparison with vanilla SDN (V-SDN), which does not use segment routing. More specifically,



(a) The multiple tunnel case.

(b) The single tunnel case.

Figure 9.4: Variation in slice admittance with change in mean demand size.

we base our performance evaluation on the following metrics: (i) slice admittance and (ii) control traffic volume. Furthermore, in line with the discussion in Section 9.4.2, we consider two variants of SR-based SDN– multi-tunnel SR-SDN (MT-SR) and single tunnel SR-SDN (ST-SR) with $\mathcal{N}_T = 1$. All simulation results are obtained using the Starlink constellation that consists of 1584 satellites at an altitude of 550 km with an inclination of $53°$.

We generate 2000 slice requests across 100 trials from a Poisson distribution with a mean rate of 1 arrival per minute, and random source destination pairings within the conterminous United States. These metrics represent an evaluation duration of over 30 hours. The primary motivation for selecting a large evaluation duration is to obtain additional insights regarding the evolution of system performance over a long period of time. The slice demands and duration values are generated from a uniform distribution and an exponential distribution respectively, while the link bandwidths (in MHz) and capacities (in Mbps) are obtained from uniform distributions of the form $U(200, 400)$ and $U(50, 1000)$ respectively. Furthermore, we set $r_{TH} = 0.7$, $|N_M| = 8$, and $\gamma = 2$.

First, in Figure 9.4, we plot the variation in demand satisfaction as the mean slice demand is varied from 25 Mbps to 100 Mbps, and the mean slice duration is varied from 15 minutes to 30 minutes. The slice admittance value represents the percentage of slices

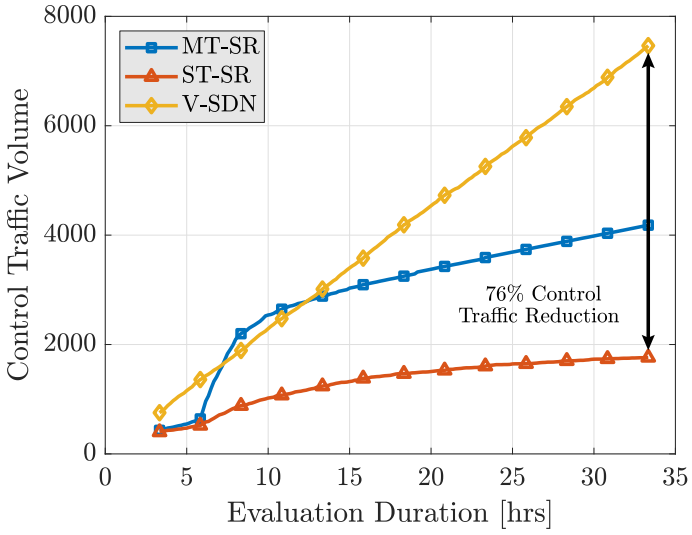


Figure 9.5: Control traffic comparison across MT-SR, ST-SR, and V-SDN.

that are admitted to the network. While Figure 9.4a compares the MT-SR case with V-SDN, Figure 9.4b presents the comparison between ST-SR and V-SDN.

From Figure 9.4 we note that, in general, slice admittance decreases as the mean slice demand and mean duration increase. This result follows from the fact that as slices consume more bandwidth and last longer, the network becomes saturated, increasing the slice rejection rate. However, there is a significant difference in performance between the SR-based solutions and V-SDN. First, from Figure 9.4a it can be seen that even in the worst case scenario MT-SR is able to accept 23% more slices than the V-SDN system, highlighting the robustness of our solution. Next, we note that while ST-SR admits fewer slices than the MT-SR approach, it still exceeds the performance the V-SDN system by 20%, as shown in Figure 9.4b. Furthermore, the difference in performance between MT-SR and ST-SR, while not significant, can be explained by the fact that MT-SR allows for the slice flows to be split across multiple tunnels, reducing the overall link utilization, and thus admitting a larger proportion of the slice requests.

Next, in Figure 9.5, we compare the control traffic volumes across MT-SR, ST-SR, and V-SDN, during the time elapsed between the first and the last slice requests. The control traffic volume represents the number of messages that are exchanged between the controller

and the CubeSats in order to successfully route the flows associated with the slices deployed in the network. The comparison in Figure 9.5 is based on a mean demand of $50\ \mathrm{Mbps}$ and a mean duration of $30\ \mathrm{minutes}$. From the figure, it is apparent that SR, in general, achieves a significant reduction in control traffic. In particular, at the end of the evaluation duration, while MT-SR achieves a 44% reduction in control traffic, ST-SR improves upon this value by achieving a 76% reduction when compared to the V-SDN system. In this case, ST-SR is able to achieve better performance than MT-SR because with ST-SR, a given slice is deployed across a single tunnel only, thereby reducing the number of CubeSats with which the controller must exchange signaling traffic.

In comparing MT-SR and ST-SR we note that while MT-SR offers slightly improved slice admittance rates, it also suffers from a significantly larger volume of control traffic. Therefore, we leverage ST-SR for the network slicing framework presented in this chapter. Not only does the choice of ST-SR make slice deployments significantly less complex, but it also allows the system to function with the least amount of control traffic possible, in exchange for a marginal reduction in the slice acceptance rate.

9.4.3 Resource Allocation

Through the online route allocation and admission control framework presented in Section 9.4.2, we are able to obtain the set of slices $\mathcal{S}' \subseteq \mathcal{S}$ that are admitted to the network during $(t-1,\ t)$, along with the values of $\alpha_{sv}, \forall\ s \in \mathcal{S}',\ v \in \mathcal{V}$ and $\alpha_{se}, \forall\ s \in \mathcal{S}',\ e \in \mathcal{E}$. In this section, we linearize some of the other aspects of the SGIN-ANS problem, with the resulting problem being referred to as SGIN-Optimal Resource Allocation (SGIN-ORA).

First, we note that the link delay, δ_e, in (9.7) can be linearized by leveraging the convex piecewise linearization technique outined in [212]. In doing so, (9.7) can be expressed as

$$\delta_e \geq m_{ez} \sum_{s \in \mathcal{S}'} \alpha_{se} c_e x_{cse} + n_{ez} \qquad \forall\ z \in \mathcal{Z}, e \in \mathcal{E}, \tag{9.32}$$

wherein the delay expression has been replaced by a convex piecewise linear approximation with $|\mathcal{Z}|$ segments. We note that while the expression in (9.32) is not an exact representation of the link delay, it is still proportional to the average network delay. The specific values associated with m_{ez} and n_{ez} are obtained in accordance with the procedure outlined in [212, §7.1]. The next non-linearity arises due to slice throughout, r_s in (9.14). Here, (9.14) can be replaced by the following set of constraints,

$$r_s \leq c_e x_{cse} \alpha_{se} + M(1 - \alpha_{se}) \qquad \forall\ s \in \mathcal{S}',\ e \in \mathcal{E}, \tag{9.33}$$

where M is a sufficiently large integer chosen so as to ensure that the slice throughput constraint in (9.33) is only enforced for those links on which the slice is deployed.

Finally, we leverage the reformulation-linearization technique (RLT) [219] to linearize (9.16), which represents the C&DH power consumption on node v due to slice s. In particular, it is the $x^3_{\mu sv}$ term that we are interested in linearizing. To this end we introduce variables $X_{\mu sv}$ and $\overline{X}_{\mu sv}\ \forall\ s \in \mathcal{S}$, $v \in \mathcal{V}$, such that,

$$X_{\mu sv} = x^2_{\mu sv} \qquad \forall\ s \in \mathcal{S},\ v \in \mathcal{V}, \tag{9.34}$$

and

$$\overline{X}_{\mu sv} = x^3_{\mu sv} = x_{\mu sv} X_{\mu sv} \qquad \forall\ s \in \mathcal{S},\ v \in \mathcal{V}. \tag{9.35}$$

Then, leveraging the bound-factor constraints for (9.34), we have

$$2\left(x_{\mu sv}\right)_L x_{\mu sv} - X_{\mu sv} \leq \left[\left(x_{\mu sv}\right)_L\right]^2, \tag{9.36}$$

$$\left[\left(x_{\mu sv}\right)_L + \left(x_{\mu sv}\right)_U\right] x_{\mu sv} - X_{\mu sv} \geq \left(x_{\mu sv}\right)_L \left(x_{\mu sv}\right)_U, \tag{9.37}$$

and,

$$2\left(x_{\mu sv}\right)_U x_{\mu sv} - X_{\mu sv} \leq \left[\left(x_{\mu sv}\right)_U\right]^2. \tag{9.38}$$

Substituting $\left(x_{\mu sv}\right)_L = 0$ and $\left(x_{\mu sv}\right)_U = 1$, we introduce the following the constraints,

$$X_{\mu sv} \geq 0 \qquad \forall\, s \in \mathcal{S},\ v \in \mathcal{V}, \tag{9.39}$$

$$X_{\mu sv} \leq x_{\mu sv} \qquad \forall\, s \in \mathcal{S},\ v \in \mathcal{V}, \tag{9.40}$$

and,

$$X_{\mu sv} \geq 2x_{\mu sv} - 1 \qquad \forall\, s \in \mathcal{S},\ v \in \mathcal{V}. \tag{9.41}$$

Similarly, the bound-factor constraints for (9.35) can be expressed as

$$\left(x_{\mu sv}\right)_L X_{\mu sv} + \left(X_{\mu sv}\right)_L x_{\mu sv} - \overline{X}_{\mu sv} \leq \left(x_{\mu sv}\right)_L \left(X_{\mu sv}\right)_L, \tag{9.42}$$

$$\left(x_{\mu sv}\right)_U X_{\mu sv} + \left(X_{\mu sv}\right)_L x_{\mu sv} - \overline{X}_{\mu sv} \geq \left(x_{\mu sv}\right)_U \left(X_{\mu sv}\right)_L, \tag{9.43}$$

$$\left(x_{\mu sv}\right)_L X_{\mu sv} + \left(X_{\mu sv}\right)_U x_{\mu sv} - \overline{X}_{\mu sv} \geq \left(x_{\mu sv}\right)_L \left(X_{\mu sv}\right)_U, \tag{9.44}$$

and,

$$\left(x_{\mu sv}\right)_U X_{\mu sv} + \left(X_{\mu sv}\right)_U x_{\mu sv} - \overline{X}_{\mu sv} \leq \left(x_{\mu sv}\right)_U \left(X_{\mu sv}\right)_U. \tag{9.45}$$

Substituting $\left(X_{\mu sv}\right)_L = 0$ and $\left(X_{\mu sv}\right)_U = 1$, we introduce the following the constraints,

$$\overline{X}_{\mu sv} \geq 0 \qquad \forall\, s \in \mathcal{S},\ v \in \mathcal{V}, \tag{9.46}$$

$$\overline{X}_{\mu sv} \leq X_{\mu sv} \qquad \forall\, s \in \mathcal{S},\ v \in \mathcal{V}, \tag{9.47}$$

$$\overline{X}_{\mu sv} \leq x_{\mu sv} \qquad \forall\, s \in \mathcal{S},\ v \in \mathcal{V}, \tag{9.48}$$

and,

$$\overline{X}_{\mu sv} \geq x_{\mu sv} + X_{\mu sv} - 1 \qquad \forall\, s \in \mathcal{S},\ v \in \mathcal{V}. \tag{9.49}$$

The SGIN-ORA problem can be then be summarized as

SGIN-Optimal Resource Allocation (SGIN-ORA)

Find: $x_{\mu sv}$ and $x_{cse} \quad \forall\, s \in \mathcal{S}',\ v \in \mathcal{V},\ e \in \mathcal{E}$

Minimize: $\mathcal{U} = \sum_{s \in \mathcal{S}} w_s \mathcal{U}_s$

Subject To: $(9.5), (9.6), (9.8), (9.10) - (9.12), (9.17), (9.19), (9.21),$
$(9.32), (9.33), (9.39) - (9.41),$ and, $(9.46) - (9.49).$

The problem presented above is a linear program (LP) and can be solved using commercial LP solvers such as CPLEX. Thus, with the SGIN-ORA problem solved, our description of the network slicing framework is complete, and we proceed to discuss and analyze the results obtained in the next section.

9.5 Results and Analyses

In this section, we present a comprehensive evaluation scenario consisting of different kinds of use case specific slices. In particular, we consider four types of use case driven slices as described in Section 9.5.1. Within this context, we examine the impact of slice deployments on the utilization of system resources, the throughput and latency metrics, and the resource distribution between different slice types. Furthermore, we also analyze the impact of slice priority on system behavior.

Table 9.2: SLA parameters for different slice types

Slice Type	**EC**	**ICB**	**RIA**	**MAR**
$\boldsymbol{\lambda_s^{SLA}}$ [pkt/s]	4000	400	6000	200
$\boldsymbol{\theta_s}$ [kbit]	1	250	20	250
$\boldsymbol{r_s^{SLA}}$ [Mbps]	4	100	120	50
$\boldsymbol{\delta_s^{SLA}}$ [ms]	150	600	300	900
$\boldsymbol{w_s}$	0.4	0.3	0.2	0.1
$\boldsymbol{w_{sr}}$	0.85	0.85	0.85	0.85
$\boldsymbol{w_{s\delta}}$	0.85	0.85	0.85	0.85
$\boldsymbol{\Gamma_s}$	0.4	0.5	0.3	0.7

9.5.1 Evaluation Scenario

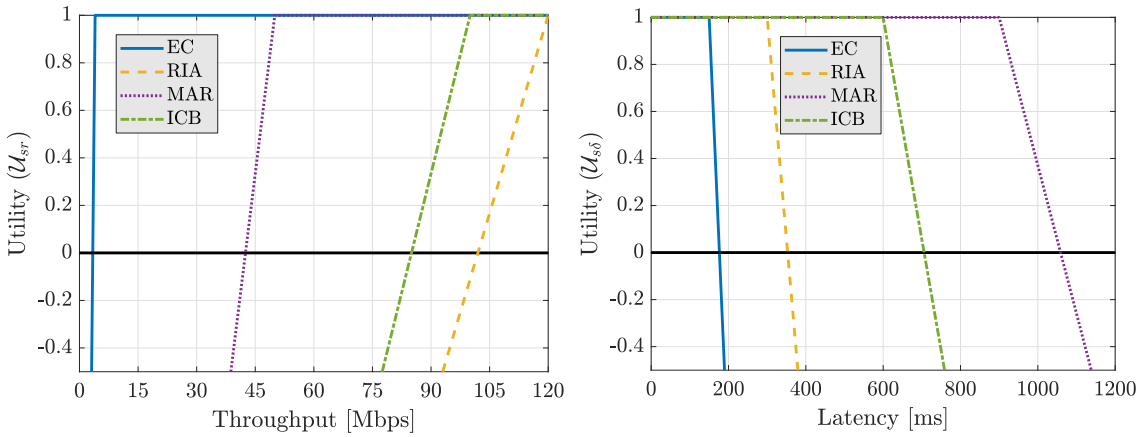
We use the Starlink constellation with 1584 satellites distributed across 24 orbital planes at an altitude of 550 km as the base network topology. The reasons for choosing Starlink are two-fold. First, it represents an ultra-dense network of satellites in line with our goals for IoST. Second, it is accompanied by a robust ground network across the conterminous United States (CONUS). Consequently, the end points for all slices under evaluation are also within CONUS. Furthermore, the non-terrestrial link bandwidths (in MHz) and terrestrial link capacities (in Mbps) are obtained from uniform distributions of the form $U(200, 400)$ and $U(50, 1000)$ respectively. The computing capacity across all CubeSats is fixed at 26500 requests per unit time with $\tau_v = 650$. In particular, we have selected these computing metrics because they translate to an effective clock speed of just under 1 GHz typically associated with on-board processors. Additionally, the power budget of all CubeSats is set to $\kappa_v = 10$ W, in keeping with typical CubeSat power budgets [143].

Building upon the use cases identified in Section 9.1, we introduce the following four types of slices: (i) emergency communications (EC), (ii) in-space cellular backhaul (ICB), (iii) remote industrial automation (RIA), and (iv) monitoring and reconnaissance (MAR). The EC use case is characterized by a low latency and low throughput operation, and,

consequently, it has the lowest SLA target latency, δ_s^{SLA}, among all slice types. On the other hand, the ICB use case is characterized by the large volume of data it must carry, and, therefore, it has the largest packet size, θ_s. Furthermore, while the RIA use case makes use of a smaller packet size, given the high frequency of industrial automation commands, an RIA slice has the highest mean packet arrival rate, λ_s^{SLA}. Additionally, latency is a critical metric for automation-related use cases, accordingly, RIA has the next lowest latency threshold after EC. Conversely, the MAR use case operates on a much longer time scale which is reflected in its high latency threshold. In fact, MAR has the highest latency threshold among all slices.

The specific SLA parameters have been outlined in Table 9.2. From the table we note that EC has a throughput-latency trade-off of $\Gamma_s = 0.4$. The lower value reflects the marginally higher importance of latency in emergency communications, on the other hand, RIA has the lowest Γ_s at 0.3, indicating the absolute importance of latency in industrial automation use cases. Furthermore, while both latency and throughput occupy center stage in the cellular backhauling use case with $\Gamma_s = 0.5$, throughput is accorded a higher priority in the MAR use case, i.e., $\Gamma_s = 0.7$. Finally, while all slices have the same throughput and latency priorities with $w_{sr} = w_{s\delta} = 0.85$, in terms of overall slice priority w_s, EC is accorded the highest priority and MAR the lowest.

Next, using the parameters presented in Table 9.2, we plot the throughput and latency utilities described in (9.18) and (9.20) respectively, in Figure 9.6. In particular, Figure 9.6a represents the throughput utility, $\mathcal{U}_{sr}$, for all slice types. In general, the utility increases from 0 at the threshold throughput, $w_{sr} r_s^{SLA}$, to 1 at the target throughput, r_s^{SLA}. However, the behavior of the utility function is customized according to the specific slice type. For example, a throughput value of $r_s = 4$ Mbps returns $\mathcal{U}_{sr} = 1$ for the EC slice type, but the same throughput would return $\mathcal{U}_{sr} < 0$ for the ICB slice, thereby reflecting the higher throughput operation that is associated with the ICB use case. In a similar vein, Figure 9.6b presents the change in latency utility, $\mathcal{U}_{s\delta}$ across all slice types. In this case, the EC slice



(a) Throughput utility.

(b) Latency utility.

Figure 9.6: Throughput and latency utility values for different slice types.

type has the steepest utility curve as a result of its strict latency requirements. On the other hand, MAR is the least impacted by an increase in network delay, reaching $\mathcal{U}_{s\delta} = 0$ only at $\delta_s = 1058$ ms.

Having presented the different slice types and their parameters along with the throughput and latency utility functions, we now examine the impact of slice deployments on the overall resource utilization, system throughput and latency, and resource distribution, along with analyzing the role of slice priority.

9.5.2 Overall System Resource Utilization

In this section we analyze the overall system resource utilization in terms of node computing, node power, and link capacity resources. The utilization levels for different system resources provide critical insight regarding both the bottlenecks that affect network performance as well the resources that are over-provisioned and consequently under-utilized. To this end, in Figure 9.7, we plot the variation in system resource utilization as the number of slices in the system, $\mathcal{N}_s$ is increased from 4 to 28. In each case, we consider an equal number of slices for each slice type, i.e., for $\mathcal{N}_s = 28$, we seek to deploy 7 slices of each type.

At the outset, we note that as the number of slices in the system increases, the average

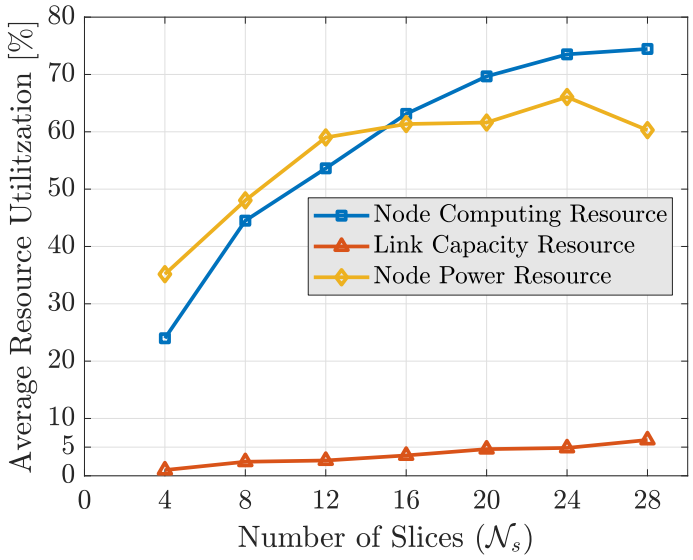


Figure 9.7: Variation in resource utilization with an increase in the number of slices.

resource resource utilization also rises. However, each of the three resource types, i.e., computing, link capacity, and power, show very different trends. First, the overall computing resource utilization hits a high of 75% in the worst case. This result is in sharp contrast to the average link utilization which barely exceeds 6% across all deployment scenarios. These results play a key role in highlighting the fact that CubeSats are compute constrained, i.e., the availability, or lack thereof, of computing resources serves as the bottleneck, while the links go largely under-utilized. To that end, the key takeaway here is that, going forward, if CubeSats are to be used for the aforementioned use cases, higher capacity on-board processors than those in use currently are going to be required.

On the other hand, concerning power consumption, we note that the utilization rises from 35% at $\mathcal{N}_s = 4$ to 66% at $\mathcal{N}_s = 24$, but then falls to 60% at $\mathcal{N}_s = 28$. This result does appear counter-intuitive at first because while the number of slices in the system has increased, the power consumption has fallen. To explain this anomaly, upon examination, we note that the overall power consumption declines due to a fall in the C&DH power consumption. As such, the C&DH power consumption solely depends on the computing resource utilization based on (9.16). Now, from Figure 9.7 we know that the average computing resource nearly

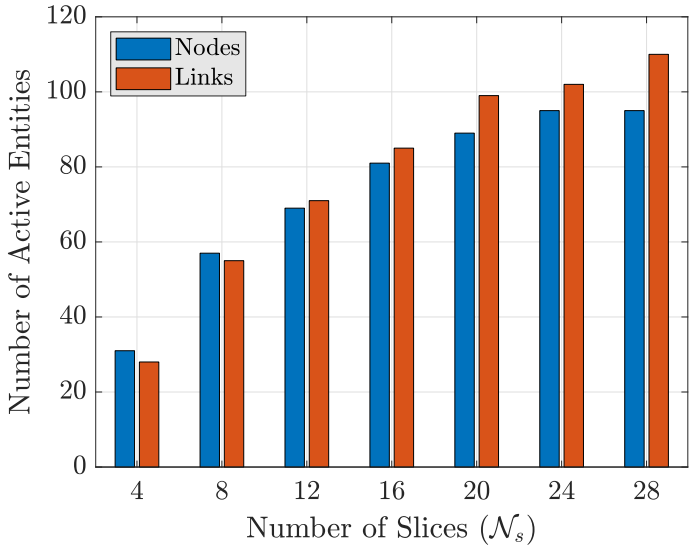
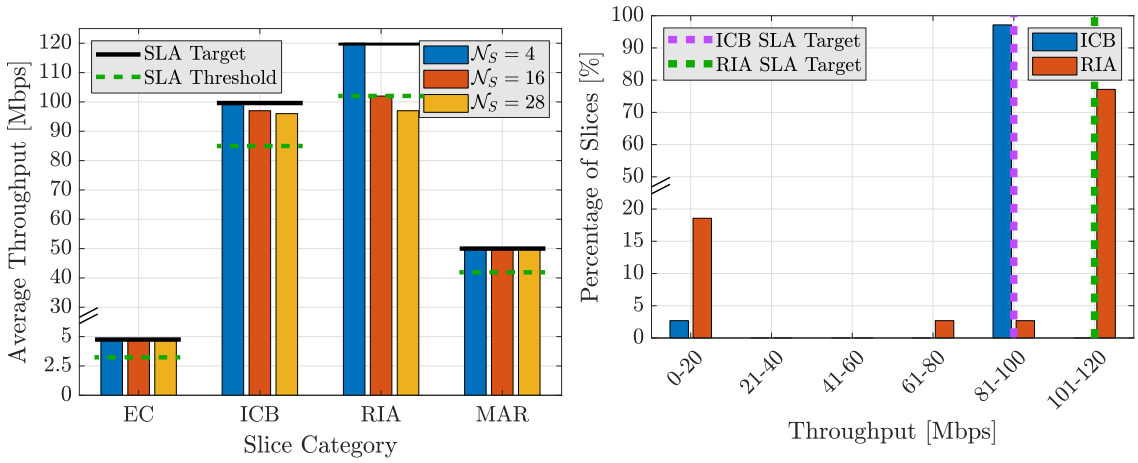


Figure 9.8: Variation in the number of active nodes and links with an increase in the number of slices.

stays at the same level of 74% in going from $\mathcal{N}_s = 24$ to $\mathcal{N}_s = 28$. Furthermore, we present the number of active nodes and links in Figure 9.8. From this figure, we can see that the number of active nodes is constant at 95 in both cases. Thus, from Figs. 9.7 and 9.8 it can be concluded that the system processes a similar number of requests per unit time in both cases. Having established that, we are interested in determining the distribution of requests by slice type, since the C&DH power consumption ultimately depends on the bits being processed by the system. As we will see later in Section 9.5.4 and Figure 9.11, as the number of slices increases, the EC slice type comes to occupy the highest proportion of computing resources. Consequently, while the system is processing a similar number of requests, there is a decline in the overall number of bits being processed, and the C&DH power consumption falls, resulting in a reduction in the overall power consumption in Figure 9.7.

9.5.3 Throughput and Latency Metrics

Having discussed the impact of slice deployments on the underlying infrastructure, we now turn our attention to key performance metrics such as throughput and latency. More specifically, in Figure 9.9a we present the average throughput grouped by slice type for



(a) Average slice throughput.

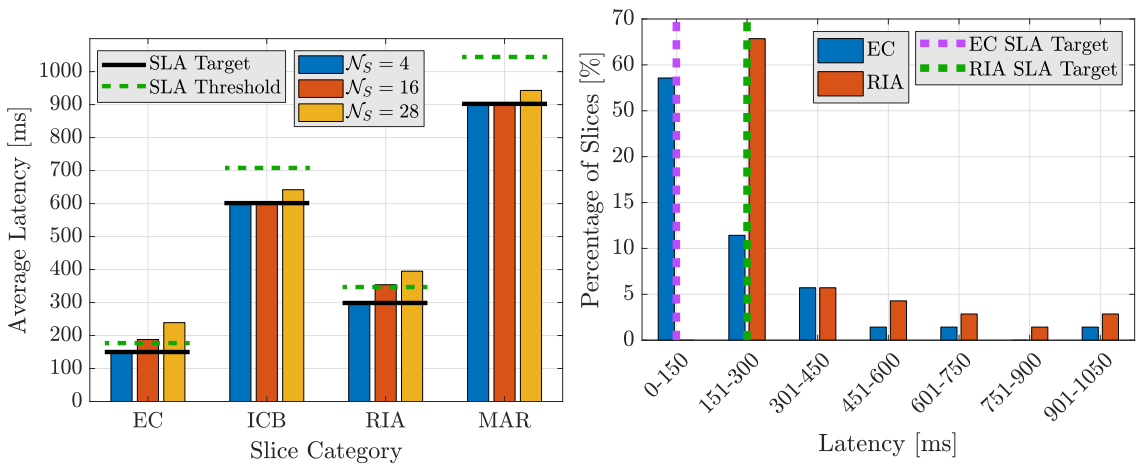
(b) Throughput distribution for ICB and RIA slice types.

Figure 9.9: Throughput metrics for different slice types.

three scenarios, i.e., $\mathcal{N}_s = 4$, $\mathcal{N}_s = 16$, and $\mathcal{N}_s = 28$. Ideally, we would like to have all slice types operating at the SLA target across all scenarios. However, as the number of slices in the system increases and resources become scarce, throughput performance falls. At the outset, the EC and MAR categories are least affected because their target throughput values are relatively lower than ICB and RIA at 4 Mbps and 50 Mbps respectively.

On the other hand, in comparing the high throughput categories of ICB and RIA, we note that while ICB operates above the threshold even in the worst case scenario, the throughput performance of RIA falls below the threshold for $\mathcal{N}_s = 28$. This result follows from the fact that with $w_s = 0.2$, RIA has a lower priority than ICB, and with $\Gamma_s = 0.3$, the RIA slice is designed to prioritize latency performance over throughput. Our assertion is further reinforced by the results showcased in Figure 9.9b, which represents the throughput distribution across all slices in the ICB and RIA categories. From the figure it can be seen that over 95% of the ICB slices operate at the SLA target, while only 75% do so for the RIA slice. Furthermore, of the proportion of RIA slices that fail to meet the target, over 15% operate in the $0 - 20$ Mbps range, indicating the lower priority and trade-off associated with the RIA category.

Next, we take a look at the latency metrics in Figure 9.10, where Figure 9.10a represents



(a) Average slice latency.

(b) Latency distribution for EC and RIA slice types.

Figure 9.10: Latency metrics for different slice types.

the average latency across all slice types and Figure 9.10b represents the latency distribution between slices in the EC and RIA categories. In general, from Figure 9.10a we can see that latency performance suffers across the board as an increasing number of slices are deployed. However, both ICB and MAR operate well within their respective latency thresholds, which is a direct outcome of their higher latency targets at 600 ms and 900 ms.

In contrast to their ICB and MAR counterparts, both EC and RIA experience SLA violations in terms of latency, owing to their significantly lower target latency values. As shown in Figure 9.10a, for $\mathcal{N}_s = 28$, EC suffers an SLA violation of 35%, exceeding its threshold latency of 177 ms by 62 ms. Similarly, RIA exceeds its threshold latency of 352 ms by 43 ms. In this case, RIA suffers a lower level of SLA violation on account of its lower trade-off value at $\Gamma_s = 0.3$ compared to EC. However, it is equally important to examine the latency distribution among slices to address the impact of overall slice priority. To this end, as shown in Figure 9.10b, the RIA slice category accounts for a larger proportion of the higher latency slices. For example, over 8% of RIA slices have a latency between $451 - 600$ ms, but this figure drops down to 2.5% for EC slices. This result can be explained by the fact that EC slices have a higher slice priority at $w_s = 0.4$, which is double that of RIA at $w_s = 0.2$. Later on, in Section 9.5.5, we will further analyze the impact of slice

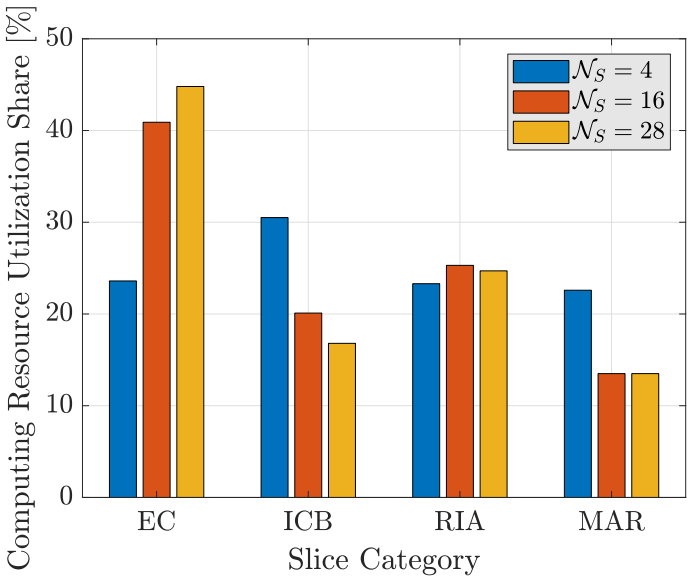


Figure 9.11: Computing resource utilization by slice category.

priority on SLA by assigning a higher priority to RIA slices.

9.5.4 Resource Distribution

The next metric we analyze is the distribution of computing resources among different slice categories as shown in Figure 9.11. We are particularly interested in the computing resource distribution because, as we have seen in Section 9.5.2, the CubeSat computing capacity serves as the bottleneck resource, unlike link capacity which is largely under-utilized. From the figure, we identify a few trends. First, as the number of slices increases, the EC and RIA slice categories come to occupy an increasingly larger share of the available computing resources, while the ICB and MAR reduce their resource occupancy. This result is a direct outcome of the fact that, with the computing resource serving as the bottleneck, the overall slice delay comes to be controlled by the share of computing resources allocated to that slice. As the EC and RIA slices have lower latency thresholds, they try to occupy a greater share of the resources to minimize their respective SLA violations.

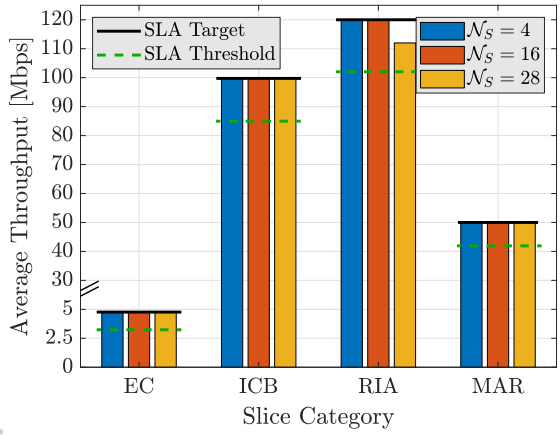
Second, in comparing EC and RIA, we note that EC is able to increase its resource occupancy at the expense of RIA as we go from $\mathcal{N}_s = 24$ to $\mathcal{N}_s = 28$. This trend can

be attributed to the higher slice priority associated with EC along with the lower latency threshold. Furthermore, these results also explain the reduction in power consumption from $\mathcal{N}_s = 24$ to $\mathcal{N}_s = 28$ as presented in Section 9.5.2, since the same amount of computing resource (in requests per unit time) is now processing fewer bits, on account of the EC category occupying nearly half the available resources.

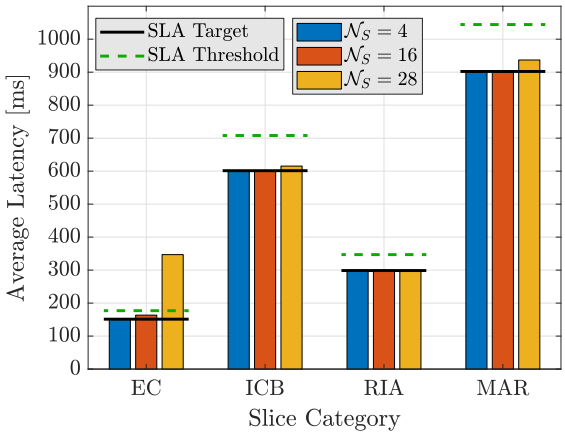
9.5.5 Slice Priority Impact

Thus far, in Sections 9.5.2 through 9.5.4 we have seen that slice priority has a direct impact on the SLA and resource utilization. The effect has been particularly pronounced in the case of RIA slices which exhibit a high throughput and relatively low latency SLA. Since the RIA slices have a lower priority than both the EC and ICB categories, throughput and latency performance suffers. To this end, in order to address the impact of slice priority, in this section, we adjust the slice priorities such that $w_s = 0.5$ for RIA, 0.2 for EC, 0.2 for ICB, and 0.1 for MAR. The other parameters are the same as outlined in Table 9.2. Under these conditions, we obtain key metrics relating to throughput, latency, and resource utilization as shown in Figure 9.12.

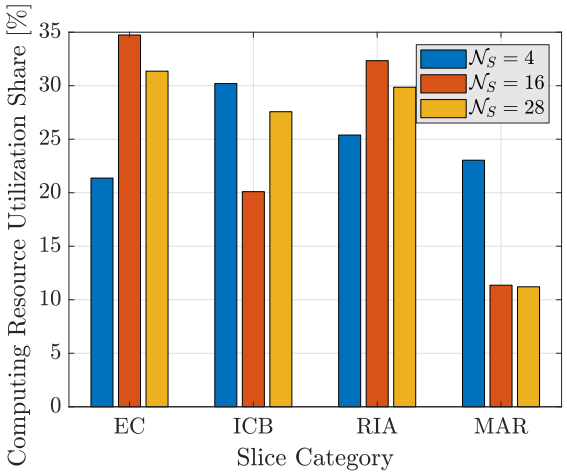
From Figure 9.12 we note that RIA is now able to meet its threshold throughput and target latency, with the lower trade-off value within RIA being responsible for the throughput falling below target. On the other hand, from Figure 9.12c, it is clear that resources are now distributed more evenly. While the lower latency target ensures that EC still occupies a marginally higher proportion of the resources at 31%, RIA now occupies 30% of the available resource pool, which is sufficient to achieve its latency targets. At the same time, latency performance for the EC category suffers greatly as result of reduced resource availability. Another interesting result is that with EC and ICB having the same priority, a portion of the resources previously accorded to EC are now occupied by the ICB slice type, allowing it to improve both its throughput and latency performance as shown in Figure 9.12a and Figure 9.12b respectively.



(a) Average slice throughput.



(b) Average slice latency.



(c) Computing resource utilization.

Figure 9.12: Performance metrics with modified slice priorities.

Furthermore, we note that the performance of the MAR category is unaffected by these changes because it still has the same priority at $w_s = 0.1$. To this end, these results help characterize the flexibility of the proposed framework and the potential for slice customization depending on changing operational scenarios. For example, during standard day-to-day operations, the EC slice type could be operated with a low priority, allowing other categories to maximize their SLA metrics. On the other hand, during emergencies EC priority can be stepped up to ensure guaranteed SLA for critical use cases. In this manner, through the results presented in Section 9.5, we note that the framework presented herein achieves our goal of automatic network slicing. In particular, through the use of automatic network

slicing, we are able to seamlessly serve slices corresponding to a variety of different use cases ranging from in-space backhauling to emergency communications. Furthermore, we are able to achieve optimal route computation and resource allocation with minimal SLA violations while also addressing the impacts of service priority.

9.6 Highlights

In this chapter we have presented a novel automatic network slicing framework for the Internet of Space Things. In doing so, we have provided a comprehensive analytical formulation of the automatic network slicing problem within the context of SGINs, addressing the dual objectives of route computation and resource allocation with minimal SLA violations. The proposed framework is characterized by a robust and flexible SLA model for use case driven slice customization. Furthermore, as part of our automatic network slicing framework, we have also introduced a novel Voronoi tessellation-based topology construction mechanism, along with a segment routing-based approach to online route computation and admission control. In addition, we have presented a comprehensive evaluation scenario with a view to benchmarking system performance in terms of resource utilization, SLA metrics, resource distribution, and service priority impacts, across a variety of use cases. The results thus obtained have demonstrated the flexibility and efficacy of the proposed framework, while also providing critical insights into resource dimensioning for ultra-dense CubeSat networks. To this end, we anticipate that the work presented herein will play a critical role in the efficient deployment of differentiated services across SGINs.

CHAPTER 10
CONTRIBUTIONS AND FUTURE RESEARCH DIRECTIONS

In this thesis, we have proposed novel SDN and NFV-based solutions for terrestrial and non-terrestrial networks with a view to realizing our vision of wireless ubiquity. The architectural frameworks and design solutions presented in this thesis have established a common network substrate geared towards a variety of applications for 6G and beyond, including but not limited to, remote healthcare, autonomous cyber-physical systems, intelligent industrial automation, precision agriculture, and smart infrastructure. The key contributions of this thesis have been summarized in the following section, followed by a discussion on the future research directions.

10.1 Contributions

In Chapter 3, we introduced a novel flexible network architecture for end-to-end cellular communications called ARBAT. In particular,

- We presented a detailed component-level description of ARBAT along with design ideas for the control plane, the data plane, and the management and orchestration solution.
- We delivered key innovations such as the Universal Network Device and Unified Cellular Network concepts, multi-slice modular resource management with the AirHYPE wireless hypervisor, network-user application interaction through the xStream platform, and simplified multi-tenant orchestration through ServiceBRIDGE.
- We conducted a comprehensive qualitative evaluation and feature comparison of ARBAT with other state-of-the-art architectures to demonstrate the advantages offered by the ARBAT platform.

In Chapter 4, we presented an optimal radio access network design framework augmented with user-specific clusters from the perspective of mobility management. In particular,

- We presented a highly accurate analytical characterization of traffic and delay-sensitive registration and paging costs.
- We developed an optimization framework that determines the optimal number of CUs, DU-CU and CU-controller assignments, along with user-specific CU clusters for signaling reduction.
- We conducted a detailed performance comparison with conventional LTE/NR networks and showed that our framework is able to achieve a reduction in signaling costs of at least 50%.

In Chapter 5, we developed a robust resource allocation framework for the containerized deployment of core network microservices across distributed cloud systems. In particular,

- We developed a mixed-integer linear programming model for container deployment to minimize overall deployment costs subject to SLA requirements associated with the microservices under consideration.
- We introduced a novel reactive event-driven framework for microservice deployment and migration.
- We conducted a comprehensive performance evaluation and showcased that our framework is able to demonstrate a more efficient utilization of resources while achieving a 30% lower cost outlay than the existing state-of-the-art solutions.

In Chapter 6, we introduced a novel cyber-physical system centered around CubeSats, known as the Internet of Space Things (IoST) with applications in monitoring and reconnaissance, in-space backhauling, and cyber-physical integration. In particular,

- We developed an end-to-end system architecture accompanied by a detailed component-level description and provided a targeted set of applications that will benefit from IoST.
- We presented the key challenges that serve as hurdles to the practical realization of IoST, and motivated possible solutions that are expected to guide research in this domain.
- We provided an initial insight into the potential of IoST through a preliminary system performance evaluation.

In Chapter 7, we presented a large-scale constellation design framework for ultra-dense CubeSat networks. In particular,

- We introduced a novel constellation coverage characterization based on the geodetic positions of CubeSats, along with spherical Voronoi tessellations, followed by CubeSat connectivity characterization based on ISL availability.
- We developed a scalable combinatorial optimization framework based on simulated annealing that provides optimal constellation configurations for a variety of use cases.
- We designed a set of constellations for IoST, each catering to a specific use case, i.e., global, latitude-specific, or region-specific, and showcased the robustness of our designs against the existing state-of-the-art.

In Chapter 8, we developed an SDN-based segment routing framework for CubeSat networks. In particular,

- We presented a robust analytical characterization of the segment routing problem along with an online algorithm for near-optimal route computation characterized by a provable performance bound.
- We conducted an exhaustive performance comparison against existing SDN-based systems to demonstrate that our framework not only ensures a higher level of demand satisfaction but also results in a significant reduction in control traffic, along with load balancing.

In Chapter 9, we presented an automatic network slicing framework for space-ground integrated networks (SGINs). In particular,

- We introduced a novel topology construction mechanism based on Voronoi tessellations to map ultra-dense constellations to equivalent network topologies.
- We developed a comprehensive analytical formulation of the automatic network slicing problem within the context of SGINs addressing the dual objectives of route computation and resource allocation with minimal SLA violations.
- We introduced robust and flexible SLA model to characterize the nuances associated with slices relating to a plethora of use cases.
- We presented a comprehensive performance evaluation of the proposed slicing framework within the context of typical application scenarios associated with IoST, demonstrating the efficacy and adaptability of our framework.

10.2 Future Research Directions

Going forward, the overarching goal is to develop autonomous programmable networks for terrestrial, high-altitude, and space communications. As networks increase in complexity, evolving to incorporate several billion endpoints across both terrestrial and space domains, network automation will play a key role in simplifying operations. To this end, we propose the following research directions for the future.

- **Stateful Data Planes for Time Evolving Networks:** Time evolving networks such as space and high-altitude systems are subject to frequent changes in network topology and typically operate in high latency environments. In such situations relying on the controller to update the forwarding state each time introduces a significant performance penalty. Instead, if the nodes performing the forwarding action are to take cognizance of the system state, route optimality could be maintained with a relatively lower control overhead. In this

research direction, we plan to address a variety of inter-related research challenges. First, we will develop a generic broad-based definition of state, along with abstractions that expose this state. Second, since state maintenance will be done by distributed switching devices, we will create a state consistency mechanism that is amenable to space and high-altitude networks. Third, we will investigate in-network computation within the context of stateful data planes. In-network computation will allow for the possibility of acquired data being processed within the space or high-altitude segment resulting in a significant reduction in latency.

- **Management and Orchestration Solutions:** The management and orchestration (MANO) framework is primarily responsible for the management of virtualized infrastructure, orchestration of network services, and the lifecycle management of network functions. Research efforts in this direction encompass both algorithmic solutions and systems design. However, there is a distinct lack of solutions tailored towards non-terrestrial networks (NTNs). To this end, we will explore service composition, service assurance, and elastic resource allocation for NTNs and SGINs by leveraging tools such as deep reinforcement learning within the context of multi-agent systems. From a systems perspective, we will extend leading MANO frameworks such as the Open Network Automation Platform (ONAP) and Open Source MANO (OSM) to incorporate support for the aforementioned solutions.
- **Self-Driving Wireless Networks and Programmability:** The increasing complexity of communications networks coupled with the constant state of flux brought forth by an ever-increasing number of connected devices has made the task of real-time network management nearly impossible for human operators. Therefore, there is a strong case for transitioning from operator-driven networks to self-driving programmable networks [220]. In this research direction, we will investigate the key enablers for self-driving programmable wireless networks. More specifically, we will develop declarative intent definitions and related parsing mechanisms tailored to cellular networks, followed by extensions to incor-

porate nanosatellite networks. Then, we will explore In-Band Network Telemetry (INT) for wireless networks. While INT is particularly suited for space networks since it does not introduce additional control traffic, it also presents several interesting challenges ranging from optimal metadata definitions and data models to operational considerations. Furthermore, drawing upon on our work in [2] and the P4 project [64], we will develop a programmable data plane for SGINs using a combination of low-level APIs and extensible protocol-specific plugins.

- **Software Platform for System Evaluation:** The operational performance of a proposed analytical solution is a practical way to gauge its efficacy. A system-level evaluation framework is thus indispensable for evaluating system performance. In recent years, the OpenAirInterface [221] and srsLTE [222] platforms have paved the way for low-cost modular testbeds within the context of cellular systems. On the other hand, OpenSAND [223] has allowed for end-to-end satellite emulation. However, these platforms are just the tip of the iceberg with much work to be done. With a view to enabling use case-driven reproducible research, we will develop an integrated evaluation framework for SGINs. More specifically, we will extend OpenAirInterface to include support for non-terrestrial access, along with developing enhancements for OpenSAND to incorporate nanosatellite networks. Next, we will integrate the two developed solutions along with a P4-based transport network. In the long term, the proposed platform will evolve to include innovations developed within these domains along with fostering contributions to the open source community. The idea here is to develop a complete end-to-end platform for SGINs that can be deployed across a variety of testbeds.

REFERENCES

[1] I. F. Akyildiz, A. Kak, and S. Nie, "6G and Beyond: The Future of Wireless Communications Systems," *IEEE Access*, vol. 8, pp. 133 995–134 030, 2020.

[2] I. Akyildiz, A. Kak, E. Khorov, A. Krasilov, and A. Kureev, "ARBAT: A flexible network architecture for QoE-aware communications in 5G systems," *Computer Networks*, vol. 147, pp. 262–279, Dec. 2018.

[3] A. Kak, A. Kureev, E. Khorov, and I. F. Akyildiz, "Radio access network design with software-defined mobility management," *Wireless Networks*, vol. 26, no. 5, pp. 3349–3362, Feb. 2020.

[4] N. D. Keni and A. Kak, "Adaptive Containerization for Microservices in Distributed Cloud Systems," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, Dec. 2020.

[5] I. F. Akyildiz and A. Kak, "The Internet of Space Things/CubeSats," *IEEE Network*, vol. 33, no. 5, pp. 212–218, Sep. 2019.

[6] A. Kak, E. Guven, U. E. Ergin, and I. F. Akyildiz, "Performance Analysis of SDN-based Internet of Space Things," in *2018 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2018.

[7] I. F. Akyildiz and A. Kak, "The Internet of Space Things/CubeSats: A ubiquitous cyber-physical system for the connected world," *Computer Networks*, vol. 150, pp. 134–149, Feb. 2019.

[8] A. Kak and I. F. Akyildiz, "Large Scale Constellation Design for the Internet of Space Things/CubeSats," in *2019 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2019.

[9] ——, "Designing Large-Scale Constellations for the Internet of Space Things With CubeSats," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1749–1768, Feb. 2021.

[10] ——, "Online Intra-domain Segment Routing for Software-defined CubeSat Networks," in *2020 IEEE Global Communications Conference (GLOBECOM)*, Jan. 2020.

[11] ——, "Towards Automatic Network Slicing for the Internet of Space Things," *submitted for journal publication*, 2021.

[12] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, "SoftCell," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies - CoNEXT '13*, 2013.

[13] H. Tullberg, P. Popovski, Z. Li, M. A. Uusitalo, A. Hoglund, O. Bulakci, M. Fallgren, and J. F. Monserrat del Rio, "The METIS 5G system concept: Meeting the 5G requirements," in *IEEE Communications Magazine*, Institute of Electrical and Electronics Engineers (IEEE), vol. 54, 2016, pp. 132–139.

[14] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, "FlexRAN," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, Dec. 2016.

[15] The 3rd Generation Partnership Project (3GPP). (2018). "Release 14."

[16] TSG-Radio Access Network, "Study of separation of NR Control Plane (CP) and User Plane (UP) for split option 2," The 3rd Generation Partnership Project (3GPP), Tech. Rep. TR 38.806, 2018, version 15.0.

[17] V. G. Vassilakis, I. D. Moscholios, B. A. Alzahrani, and M. D. Logothetis, "A software-defined architecture for next-generation cellular networks," in *2016 IEEE International Conference on Communications (ICC)*, May 2016.

[18] H. Wang, S. Chen, H. Xu, M. Ai, and Y. Shi, "SoftNet: A software defined decentralized mobile network architecture toward 5G," *IEEE Network*, vol. 29, no. 2, pp. 16–22, 2015.

[19] A. Gopalasingham, L. Roullet, N. Trabelsi, C. S. Chen, A. Hebbar, and E. Bizouarn, "Generalized software defined network platform for radio access networks," in *IEEE Consumer Communications and Networking Conference (CCNC)*, 2016.

[20] "Creating an ecosystem for vRANs supporting non-ideal fronthaul," Telecom Infra Project, Tech. Rep., 2019.

[21] "NFV C-RAN for Efficient RAN Resource Allocation," NEC Corporation, Tech. Rep., 2016.

[22] The xRAN Alliance. (2018). "xRAN Forum."

[23] S. Singh and A. Raja, "xRAN Reference Controller Design Specification," The Open Networking Foundation (ONF), Tech. Rep., 2017.

[24] xRAN Fronthaul Working Group, "Control, User and Synchronization Plane Specification," xRAN.org, Tech. Rep. XRAN-FH.CUS.0, 2018, version 2.0.

[25] "Central Office Re-architected as a Datacenter (CORD)," Open Networking Foundation (ONF), Tech. Rep., 2016.

[26] "M-CORD as an Open Reference Solution for 5G Enablement," Open Networking Foundation (ONF), Tech. Rep., 2017.

[27] NGMN Alliance, "5G End-to-end Architecture Framework," Tech. Rep., 2017.

[28] A. Banchs, M. Breitbach, X. Costa, U. Doetsch, S. Redana, C. Sartori, and H. Schotten, "A Novel Radio Multiservice Adaptive Network Architecture for 5G Networks," in *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, May 2015.

[29] J. Gutiérrez, N. Maletic, D. Camps-Mur, E. Garcia, I. Berberana, M. Anastasopoulos, A. Tzanakaki, V. Kalokidou, P. Flegkas, D. Syrivelis, *et al.*, "5G-XHaul: a converged optical and wireless solution for 5G transport networks," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 9, pp. 1187–1195, 2016.

[30] F. Khan, "Mobile Internet from the Heavens," Aug. 9, 2015. arXiv: http://arxiv.org/abs/1508.02383v1 [cs.NI].

[31] Business Wire, *Astrocast Launches First Test Satellite of IoT CubeSat Network*, 2018.

[32] Space News, *Australian startup Fleet prepares Series A to fund next 10 satellites*, 2019.

[33] Kepler Communications, *Kepler's first Ku-band satellite is in orbit*, Jan. 2018.

[34] Via Satellite, *Aerial & Maritime, Aistech Enter Mutual Data Service Agreement*, Mar. 2017.

[35] "Myriota Product Brief," Myriota, Tech. Rep., 2019.

[36] Planet Labs, *Planet Launches Satellite Constellation to Image the Whole Planet Daily*, 2017.

[37] Lacuna Space, *About – Lacuna*, 2019.

[38] L. P. Dyrud, R. La Tour, W. H. Swartz, S. Nag, S. R. Lorentz, T. Hilker, W. J. Wiscombe, and S. J. Papadakis, "The power of inexpensive satellite constellations," in *Micro-and Nanotechnology Sensors, Systems, and Applications VI*, International Society for Optics and Photonics, vol. 9083, 2014, 90832A.

[39] TechCrunch, *FCC approves SpaceX plan for 4,425-satellite broadband network*, 2018.

[40] G. Shotwell. (2018). “SpaceX’s plan to fly you across the globe in 30 minutes.”

[41] “Kepler Non-Geostationary Satellite System,” Kepler Communications, Inc., Tech. Rep., 2017.

[42] J. Bao, B. Zhao, W. Yu, Z. Feng, C. Wu, and Z. Gong, “OpenSAN,” in *Proceedings of the 2014 ACM conference*, 2014.

[43] M. Sheng, Y. Wang, J. Li, R. Liu, D. Zhou, and L. He, “Toward a flexible and reconfigurable broadband satellite network: Resource management architecture and strategies,” *IEEE Wireless Communications*, vol. 24, no. 4, pp. 127–133, Aug. 2017.

[44] R. Ferrús, H. Koumaras, O. Sallent, G. Agapiou, T. Rasheed, M.-A. Kourtis, C. Boustie, P. Gélard, and T. Ahmed, “SDN/NFV-enabled satellite communications networks: Opportunities, scenarios and challenges,” *Physical Communication*, vol. 18, pp. 95–112, Mar. 2016.

[45] L. Bertaux, S. Medjiah, P. Berthou, S. Abdellatif, A. Hakiri, P. Gelard, F. Planchou, and M. Bruyere, “Software defined networking and virtualization for broadband satellite networks,” *IEEE Communications Magazine*, vol. 53, no. 3, pp. 54–60, Mar. 2015.

[46] T. Ahmed, E. Dubois, J.-B. Dupé, R. Ferrús, P. Gélard, and N. Kuhn, “Software-defined satellite cloud RAN,” *International Journal of Satellite Communications and Networking*, vol. 36, no. 1, pp. 108–133, Feb. 2017.

[47] T. Li, H. Zhou, H. Luo, Q. Xu, and Y. Ye, “Using SDN and NFV to implement satellite communication networks,” in *2016 International Conference on Networking and Network Applications (NaNA)*, Jul. 2016.

[48] T. Li, H. Zhou, H. Luo, and S. Yu, “SERvICE: A Software Defined Framework for Integrated Space-Terrestrial Satellite Communication,” *IEEE Transactions on Mobile Computing*, vol. 17, no. 3, pp. 703–716, Mar. 2018.

[49] ——, “SERvICE: A software defined framework for integrated space-terrestrial satellite communication,” *IEEE Transactions on Mobile Computing*, pp. 1–1, 2017.

[50] P. Du, S. Nazari, J. Mena, R. Fan, M. Gerla, and R. Gupta, “Multipath TCP in SDN-enabled LEO satellite networks,” in *MILCOM 2016 - 2016 IEEE Military Communications Conference*, IEEE, Nov. 2016.

[51] T. Li, H. Zhou, H. Luo, W. Quan, and S. Yu, “Modeling software defined satellite networks using queueing theory,” in *2017 IEEE International Conference on Communications (ICC)*, IEEE, May 2017.

[52] S. Nazari, P. Du, M. Gerla, C. Hoffmann, J. H. Kim, and A. Capone, "Software defined naval network for satellite communications (SDN-SAT)," in *MILCOM 2016 - 2016 IEEE Military Communications Conference*, IEEE, Nov. 2016.

[53] B. Yang, Y. Wu, X. Chu, and G. Song, "Seamless handover in software-defined satellite networking," *IEEE Communications Letters*, vol. 20, no. 9, pp. 1768–1771, Sep. 2016.

[54] M. D. Sanctis, E. Cianca, G. Araniti, I. Bisio, and R. Prasad, "Satellite Communications Supporting Internet of Remote Things," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 113–123, Feb. 2016.

[55] V. Almonacid and L. Franck, "Extending the coverage of the internet of things with low-cost nanosatellite networks," *Acta Astronautica*, vol. 138, pp. 95–101, Sep. 2017.

[56] The 3rd Generation Partnership Project (3GPP). (2019). "Release 15."

[57] "Cisco Visual Networking Index," Cisco Systems, Inc., Tech. Rep., 2017.

[58] "Framework and overall objectives of the future development of IMT for 2020 and beyond," International Telecommunication Union (ITU), Tech. Rep. RM.2083-0, Sep. 2015.

[59] A. Sorsaniemi, "5G and Energy Efficiency," DG Connect, EC, Tech. Rep., Nov. 2017.

[60] L. Duan, J. Huang, and J. Walrand, "Economic analysis of 4G network upgrade," in *2013 Proceedings IEEE INFOCOM*, Apr. 2013.

[61] I. F. Akyildiz, S. Nie, S.-C. Lin, and M. Chandrasekaran, "5G roadmap: 10 key enabling technologies," *Computer Networks*, vol. 106, pp. 17–48, Sep. 2016.

[62] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, Apr. 2014.

[63] "Network Functions Virtualisation," European Telecommunications Standards Institute (ETSI), Tech. Rep., 2012.

[64] P. Bosshart, G. Varghese, D. Walker, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, and A. Vahdat, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, Jul. 2014.

[65] TSG-Services and System Aspects, "Study on management and orchestration of network slicing for next generation network," The 3rd Generation Partnership Project (3GPP), Tech. Rep. TR 28.801, 2018, version 15.1.

[66] J. Aweya, "IP router architectures: An overview," *International Journal of Communication Systems*, vol. 14, no. 5, pp. 447–475, 2001.

[67] TSG– Services and System Aspects, "System Architecture for the 5G System," The 3rd Generation Partnership Project (3GPP), Tech. Rep. TS 23.501, 2018, version 15.2.

[68] M. Jaber, M. A. Imran, R. Tafazolli, and A. Tukmanov, "5G Backhaul Challenges and Emerging Research Directions: A Survey," *IEEE Access*, vol. 4, pp. 1743–1766, 2016.

[69] B. Guo, W. Cao, A. Tao, and D. Samardzija, "LTE/LTE-a signal compression on the CPRI interface," *Bell Labs Technical Journal*, vol. 18, no. 2, pp. 117–133, Sep. 2013.

[70] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "KVM: the Linux Virtual Machine Monitor," in *In Proceedings of the 2007 Ottawa Linux Symposium (OLS'-07)*, 2007.

[71] M. Helsley, "LXC: Linux container tools," *IBM devloperWorks Technical Library*, vol. 11, 2009.

[72] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proceedings of the nineteenth ACM symposium on Operating systems principles - SOSP '03*, 2003.

[73] J. A. Kappel, A. Velte, and T. Velte, *Microsoft Virtualization with Hyper-V: Manage Your Datacenter with Hyper-V, Virtual PC, Virtual Server, and Application Virtualization*. McGraw-Hill, Inc., Aug. 11, 2009, 430 pp., ISBN: 0071614036.

[74] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. Mckeown, and G. Parulkar, "FlowVisor: A network virtualization layer," OpenFlow Consortium, Tech. Rep., 2009.

[75] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on Network Virtualization Hypervisors for Software Defined Networking," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 655–685, 2016.

[76] D. Hancock and J. van der Merwe, "HyPer4," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies - CoNEXT'16*, 2016.

[77] C. Zhang, J. Bi, Y. Zhou, A. B. Dogar, and J. Wu, "HyperV: A High Performance Hypervisor for Virtualization of the Programmable Data Plane," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, Jul. 2017.

[78] M. Kalil, M. Youssef, A. Shami, A. Al-Dweik, and S. Ali, "Wireless resource virtualization: Opportunities, challenges, and solutions," *Wireless Communications and Mobile Computing*, vol. 16, no. 16, pp. 2690–2699, Jul. 2016.

[79] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network Slicing in 5G: Survey and Challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, May 2017.

[80] "PDU Session User Plane protocol," The 3rd Generation Partnership Project (3GPP), Tech. Rep. 38.415, 2018, version 15.1.0.

[81] G. Cheng, H. Chen, H. Hu, Z. Wang, and J. Lan, "Enabling network function combination via service chain instantiation," *Computer Networks*, vol. 92, pp. 396–407, Dec. 2015.

[82] TSG-Services and System Aspects, "Study on charging aspects of 5G system architecture phase 1," The 3rd Generation Partnership Project (3GPP), Tech. Rep. TR 32.899, 2018, version 15.0.

[83] 5G Innovation Center, "The Flat Distributed Cloud (FDC) 5G Architecture Revolution," Institute for Communication Systems, University of Surrey, Tech. Rep., Jan. 2016.

[84] I. F. Akyildiz, E. Khorov, A. Kiryanov, D. Kovkov, A. Krasilov, M. Liubogoshchev, D. Shmelkin, and S. Tang, "xStream: A New Platform Enabling Communication between Applications and the 5G Network," in *2018 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2018.

[85] H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: Myths, caveats, and the best practices," in *Proceedings of the 2008 ACM CoNEXT conference*, 2008, p. 11.

[86] O. Sallent, J. Perez-Romero, R. Ferrus, and R. Agusti, "On Radio Access Network Slicing from a Radio Resource Management Perspective," *IEEE Wireless Communications*, vol. 24, no. 5, pp. 166–174, Oct. 2017.

[87] M. Richart, J. Baliosian, J. Serrat, and J.-L. Gorricho, "Resource Slicing in Virtual Wireless Networks: A Survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 462–476, Sep. 2016.

[88] "Network Functions Virtualisation (NFV); Management and Orchestration," European Telecommunications Standards Institute (ETSI), Tech. Rep. ETSI GS NFV-MAN 001, 2014, version 1.1.1.

[89] "Architecture Overview," Open Network Automation Platform (ONAP), Tech. Rep., 2018.

[90] Open Source MANO, "OSM Release FOUR Technical Overview," European Telecommunications Standards Institute (ETSI), Tech. Rep., 2018.

[91] METIS-II, "Final Overall 5G RAN Design," The 5G Infrastructure Public Private Partnership (5G-PPP), Tech. Rep., 2017.

[92] "O-RAN: Towards an Open and Smart RAN," O-RAN Alliance, Tech. Rep., 2019.

[93] F. Giust, L. Cominardi, and C. Bernardos, "Distributed mobility management for future 5G networks: overview and analysis of existing approaches," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 142–149, Jan. 2015.

[94] I. Ahmad, M. Liyanage, S. Namal, M. Ylianttila, A. Gurtov, M. Eckert, T. Bauschert, Z. Faigl, L. Bokor, E. Saygun, H. A. Akyildiz, O. L. Perez, M. U. Itzazelaia, B. Ozbek, and A. Ulas, "New concepts for traffic, resource and mobility management in software-defined mobile networks," *2016 12th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pp. 1–8, 2016.

[95] T.-T. Nguyen, C. Bonnet, and J. Harri, "SDN-based distributed mobility management for 5G networks," in *2016 IEEE Wireless Communications and Networking Conference*, Apr. 2016.

[96] L. M. Contreras, L. Cominardi, H. Qian, and C. J. Bernardos, "Software-Defined Mobility Management: Architecture Proposal and Future Directions," *Mobile Networks and Applications*, vol. 21, no. 2, pp. 226–236, Jan. 2016.

[97] K. Tantayakul, R. Dhaou, and B. Paillassa, "Impact of SDN on Mobility Management," in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, Mar. 2016.

[98] Y. Kim, S. M. Raza, D. T. Nguyen, S. Jeon, and H. Choo, "Towards On-Demand Mobility Management in SDN," in *Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication*, 2018.

[99] X. Yin, L. Wang, and S. Jiang, "A hierarchical mobility management scheme based on software defined networking," *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 310–325, Jan. 2018.

[100] A. S. D. Alfoudi, S. H. S. Newaz, R. Ramlie, G. M. Lee, and T. Baker, "Seamless Mobility Management in Heterogeneous 5G Networks: A Coordination Approach among Distributed SDN Controllers," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, Apr. 2019.

[101] R. M. Alaez, E. Chirivella-Perez, J. M. A. Calero, and Q. Wang, "New Topology Management Scheme in LTE and 5G Networks," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, Jun. 2018.

[102] R. Mijumbi, J. Serrat, J.-L. Gorricho, J. Rubio-Loyola, and S. Davy, "Server placement and assignment in virtualized radio access networks," in *2015 11th International Conference on Network and Service Management (CNSM)*, Nov. 2015.

[103] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-optimal resilient controller placement in SDN-based core networks," in *Proceedings of the 2013 25th International Teletraffic Congress (ITC)*, Sep. 2013.

[104] S. Guo, S. Yang, Q. Li, and Y. Jiang, "Towards Controller Placement for robust Software-Defined Networks," in *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, Dec. 2015.

[105] Q. Zhong, Y. Wang, W. Li, and X. Qiu, "A min-cover based controller placement approach to build reliable control network in SDN," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, Apr. 2016.

[106] T. Wang, F. Liu, and H. Xu, "An Efficient Online Algorithm for Dynamic SDN Controller Assignment in Data Center Networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2788–2801, Oct. 2017.

[107] L. Liao and V. C. M. Leung, "Genetic algorithms with particle swarm optimization based mutation for distributed controller placement in SDNs," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov. 2017.

[108] M. Tanha, D. Sajjadi, R. Ruby, and J. Pan, "Capacity-Aware and Delay-Guaranteed Resilient Controller Placement for Software-Defined WANs," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 991–1005, Sep. 2018.

[109] S.-C. Lin, P. Wang, I. F. Akyildiz, and M. Luo, "Towards Optimal Network Planning for Software-Defined Networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 12, pp. 2953–2967, Dec. 2018.

[110] T. Janevski, *Traffic Analysis and Design of Wireless IP Networks*. Artech House Publishers, 2003.

[111] G. Strang, *Introduction to Linear Algebra*. Cambridge University Press, Aug. 11, 2016, 600 pp., ISBN: 0980232775.

[112] W. Willinger, V. Paxson, R. Riedi, and M. Taqqu, "Long-range dependence and data network traffic," *Long-range Dependence: Theory and Applications*, 2002.

[113] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 1–15, 1994.

[114] M. Zukerman, T. Neame, and R. Addie, "Internet traffic modeling and future technology implications," in *2003 IEEE Conference on Computer Communications (INFOCOM)*.

[115] The MathWorks, Inc., *MATLAB R2019a*, 2019.

[116] A. Wu, "Taking the Cloud-Native Approach with Microservices," Magenic/Google Cloud Platform, Tech. Rep., 2017.

[117] M. Amundsen and M. Mclarty, *Microservice Architecture*. O'Reilly Media, Inc, USA, Aug. 19, 2016, 146 pp., ISBN: 1491956259.

[118] J. B. Moreira, H. Mamede, V. Pereira, and B. Sousa, "Next generation of microservices for the 5G Service-Based Architecture," *International Journal of Network Management*, vol. 30, no. 6, Aug. 2020.

[119] R. Morabito, J. Kjallman, and M. Komu, "Hypervisors vs. Lightweight Virtualization: A Performance Comparison," in *2015 IEEE International Conference on Cloud Engineering*, Mar. 2015.

[120] D.-N. Le, R. Kumar, G. N. Nguyen, and J. M. Chatterjee, *Cloud Computing and Virtualization*. John Wiley & Sons, Mar. 12, 2018, 234 pp.

[121] C. Pahl, "Containerization and the PaaS cloud," *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, May 2015.

[122] M. Aazam and E.-N. Huh, "Cloud broker service-oriented resource management model," *Transactions on Emerging Telecommunications Technologies*, vol. 28, no. 2, pp. 2937–2953, Apr. 2015.

[123] S. S. Chauhan, E. S. Pilli, R. Joshi, G. Singh, and M. Govil, "Brokering in interconnected cloud computing environments: A survey," *Journal of Parallel and Distributed Computing*, Aug. 2018.

[124] J. Monsalve, A. Landwehr, and M. Taufer, "Dynamic CPU Resource Allocation in Containerized Cloud Environments," in *2015 IEEE International Conference on Cluster Computing*, Sep. 2015.

[125] X. Wan, X. Guan, T. Wang, G. Bai, and B.-Y. Choi, "Application deployment using Microservice and Docker containers: Framework and optimization," *Journal of Network and Computer Applications*, vol. 119, pp. 97–109, Oct. 2018.

[126] C. Guerrero, I. Lera, and C. Juiz, "Genetic Algorithm for Multi-Objective Optimization of Container Allocation in Cloud Architecture," *Journal of Grid Computing*, vol. 16, no. 1, pp. 113–135, Nov. 2017.

[127] M. Nardelli, C. Hochreiner, and S. Schulte, "Elastic Provisioning of Virtual Machines for Container Deployment," in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion - ICPE '17 Companion*, 2017.

[128] T. Zheng, X. Zheng, Y. Zhang, Y. Deng, E. Dong, R. Zhang, and X. Liu, "SmartVM: A SLA-aware microservice deployment framework," *World Wide Web*, vol. 22, no. 1, pp. 275–293, May 2018.

[129] S. Chen, C. Delimitrou, and J. F. Martinez, "PARTIES," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS '19*, 2019.

[130] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf, "NIST Cloud Computing Reference Architecture," Tech. Rep., 2011.

[131] Docker, Inc. (2018). "Docker Swam Scheduling Strategies."

[132] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*. Springer-Verlag Berlin Heidelberg, 2006.

[133] "State of the Market: Internet of Things 2017," Verizon, Tech. Rep., 2017.

[134] D. Lund, C. MacGillivray, V. Turner, and M. Morales, "Worldwide and Regional Internet of Things (IoT) 2014–2020 Forecast: A Virtuous Circle of Proven Value and Demand," International Data Corporation, Tech. Rep., 2014.

[135] L. Wood, "Supporting group applications via satellite constellations with multicast," in *Sixth IEE Conference on Telecommunications*, IEE, 1998.

[136] Z. Qu, G. Zhang, H. Cao, and J. Xie, "LEO satellite constellation for internet of things," *IEEE Access*, vol. 5, pp. 18 391–18 401, 2017.

[137] L. A. Davis and L. Filip, "How long does it take to develop and launch government satellite systems?" The Aerospace Corporation, Tech. Rep., 2015.

[138] W. A. Hanson, "Satellite internet in the mobile age," *New Space*, vol. 4, no. 3, pp. 138–152, Sep. 2016.

[139] S. Madry, *Space Systems for Disaster Warning, Response, and Recovery (SpringerBriefs in Space Development)*. Springer, 2014, ISBN: 9781493915125.

[140] W. Ailor, G. Peterson, J. Womack, and M. Youngs, "Effect of large constellations on lifetime of satellites in low earth orbits," *Journal of Space Safety Engineering*, vol. 4, no. 3-4, pp. 117–123, Sep. 2017.

[141] R. P. Welle, "The Cubesat Paradigm: An Evolutionary Approach To Satellite Design," The Aerospace Corporation, Tech. Rep., 2016.

[142] "Hitomi Experience Report: Investigation of Anomalies Affecting the X-ray Astronomy Satellite "Hitomi" (ASTRO-H)," Japan Aerospace Exploration Agency, Tech. Rep., 2016.

[143] I. F. Akyildiz, J. M. Jornet, and S. Nie, "A new CubeSat design with reconfigurable multi-band radios for dynamic spectrum satellite communication networks," *Ad Hoc Networks*, vol. 86, pp. 166–178, Apr. 2019.

[144] K. Woellert, P. Ehrenfreund, A. J. Ricco, and H. Hertzfeld, "Cubesats: Cost-effective science and technology platforms for emerging and developing nations," *Advances in Space Research*, vol. 47, no. 4, pp. 663–684, 2011.

[145] I. F. Akyildiz, J. M. Jornet, and S. Nie, "A New CubeSat Design with Reconfigurable Multi-Band Radios for Communication in Dynamic Spectrum Frequencies," submitted for journal publication, May 2018.

[146] Jet Propulsion Laboratory, "Mars Cube One (MarCO) Fact Sheet," National Aeronautics and Space Administration (NASA), Tech. Rep., 2018.

[147] M. A. Swartwout, "CubeSats and Mission Success: 2017 Update," *2017 Electronic Technology Workshop, NASA Electronic Parts and Packaging Program (NEPP)*, Jun. 2017.

[148] C. Nagarajan, R. G. D'souza, S. Karumuri, and K. Kinger, "Design of a cubesat computer architecture using COTS hardware for terrestrial thermal imaging," in *2014 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology*, IEEE, Nov. 2014.

[149] C. Robson, "Comparison of CubeSats, CubeSat Swarms and Classical Earth Observation Satellites in LEO," *Canadian SmallSat Symposium*, Feb. 2016.

[150] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, "Wireless sensor network virtualization: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 553–576, 2016.

[151] E. Baccelli, O. Hahm, M. Gunes, M. Wahlisch, and T. Schmidt, "RIOT OS: Towards an OS for the Internet of Things," in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, Apr. 2013.

[152] A. D. Santangelo, "An Open Source Space Hypervisor For Small Satellites," in *AIAA SPACE 2013 Conference and Exposition*, American Institute of Aeronautics and Astronautics, Sep. 2013.

[153] "Sdn architecture," Open Networking Foundation (ONF), Tech. Rep. TR-521, 2016, Issue 1.1.

[154] P. Goransson and C. Black, *Software Defined Networks: A Comprehensive Approach*. Morgan Kaufmann, 2014, ISBN: 978-0124166752.

[155] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, "The Segment Routing Architecture," in *2015 IEEE Global Communications Conference (GLOBECOM)*, IEEE, Dec. 2015.

[156] S. M. Guertin, "Candidate Cubesat Processors," in *NEPP EEE Parts for Small Missions Workshop*, 2014.

[157] K. Suo, Y. Zhao, W. Chen, and J. Rao, "An Analysis and Empirical Study of Container Networks," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, IEEE, Apr. 2018.

[158] A. Guidotti, A. Vanelli-Coralli, O. Kodheli, G. Colavolpe, and T. Foggi, "Integration of 5G Technologies in LEO Mega-Constellations," Sep. 18, 2017, arXiv:1709.05807. arXiv: http://arxiv.org/abs/1709.05807v1 [cs.NI].

[159] *Systems Tool Kit – AGI*, 2020.

[160] R. Elitzur, T. R. Gurlitz, S. B. Sedlacek, and K. H. Senechal, "ASTROLIB Users Guide," Navigation and Geopositioning Systems Department, The Aerospace Corporation, Tech. Rep. 93-(3917-1), 1993.

[161] "Aerospace Capabilities and Tools for Advancing Small Launch," The Aerospace Corporation, Tech. Rep., 2018.

[162] S. Nag, S. P. Hughes, and J. L. Moigne, "Streamlining the Design Tradespace for Earth Imaging Constellations," in *AIAA SPACE 2016*, Sep. 2016.

[163] N. H. Crisp, S. Livadiotti, and P. C. E. Roberts, "A Semi-Analytical Method for Calculating Revisit Time for Satellite Constellations with Discontinuous Coverage," Jul. 5, 2018. arXiv: 1807.02021v1 [cs.CE].

[164] NASA GSFC, *General Mission Analysis Tool (GMAT)*, 2018.

[165] J. L. C. Rodríguez, "poliastro Documentation," poliastro Development Team, Tech. Rep., 2018, Release 0.11.1.

[166] JuliaSpace, *Satellite Toolbox*, 2019.

[167] S. Nag, J. L. Rios, D. Gerhardt, and C. Pham, "CubeSat constellation design for air traffic monitoring," *Acta Astronautica*, vol. 128, pp. 180–193, Nov. 2016.

[168] A. Marinan, A. Nicholas, and K. Cahoy, "Ad hoc CubeSat constellations: Secondary launch coverage and distribution," in *2013 IEEE Aerospace Conference*, IEEE, Mar. 2013.

[169] W. R. Whittecar and M. P. Ferringer, "Global Coverage Constellation Design Exploration Using Evolutionary Algorithms," in *AIAA/AAS Astrodynamics Specialist Conference*, Aug. 2014.

[170] T. Savitri, Y. Kim, S. Jo, and H. Bang, "Satellite Constellation Orbit Design Optimization with Combined Genetic Algorithm and Semianalytical Approach," *International Journal of Aerospace Engineering*, vol. 2017, pp. 1–17, 2017.

[171] S. Liu, P. Li, G. Cui, and W. Wang, "Design of satellite constellation with inter-satellite links for global communication using genetic algorithm," in *2017 20th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, Dec. 2017.

[172] N. Hitomi and D. Selva, "Constellation optimization using an evolutionary algorithm with a variable-length chromosome," in *2018 IEEE Aerospace Conference*, Mar. 2018.

[173] W. Su, J. Lin, and T. Ha, "Global communication coverage using Cubesats," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, Jan. 2017.

[174] K. Hofner, A. Vahl, and E. Stoll, "Cost-efficient satellite constellation design by network reliability analysis," in *2018 Annual Reliability and Maintainability Symposium (RAMS)*, Jan. 2018.

[175] O. P. Gupta, "Iridium NEXT SensorPODs: Global access for your scientific payloads," in *Proceedings of the 25th Annual AIAA/USU Conference on Small Satellites*, Aug. 2011.

[176] "OneWeb Non-Geostationary Satellite System," OneWeb, Tech. Rep., 2018.

[177] Kuiper Systems, *Application for Fixed Satellite Service by Kuiper Systems LLC*, 2019.

[178] D. A. Vallado, *Fundamentals of Astrodynamics and Applications*, Fourth, J. Wertz, Ed. Microcosm Press, 2013.

[179] Office of Geomatics, *World Geodetic System 1984 (WGS 84)*.

[180] K. Osen, "Accurate Conversion of Earth-Fixed Earth-Centered Coordinates to Geodetic Coordinates," Norwegian University of Science and Technology, Tech. Rep., 2017, hal-01704943.

[181] R. J. Renka, "Algorithm 772: STRIPACK: Delaunay triangulation and Voronoi diagram on the surface of a sphere," *ACM Transactions on Mathematical Software*, vol. 23, no. 3, pp. 416–434, Sep. 1997.

[182] R. C. Burk, "Closed-Form Approximation of Revisit Rate for Low-Altitude Satellites," *Journal of Spacecraft and Rockets*, vol. 50, no. 4, pp. 872–883, Jul. 2013.

[183] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[184] E. H. Aarts and P. J. van Laarhoven, *Simulated Annealing: Theory and Applications*. Springer Netherlands, Jun. 30, 1987, 204 pp., ISBN: 9027725136.

[185] M. Srinivas and L. Patnaik, "Genetic algorithms: A survey," *Computer*, vol. 27, no. 6, pp. 17–26, Jun. 1994.

[186] H. A. e Oliveira Junior, L. Ingber, A. Petraglia, M. R. Petraglia, and M. A. S. Machado, "Adaptive Simulated Annealing," in *Intelligent Systems Reference Library*, Springer Berlin Heidelberg, 2012, pp. 33–62.

[187] L. Ingber and B. Rosen, "Genetic Algorithms and Very Fast Simulated Reannealing: A comparison," *Mathematical and Computer Modelling*, vol. 16, no. 11, pp. 87–100, Nov. 1992.

[188] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, Jun. 1953.

[189] Y. F. Wong, O. Kegege, S. H. Schaire, G. Bussey, and S. Altunc, "An Optimum Space-to-Ground Communication Concept for CubeSat Platform Utilizing NASA Space Network and Near Earth Network," in *Small Satellite Conference*, 2016.

[190] K. Devaraj, M. Ligon, E. Blossom, J. Breu, B. Klofas, K. Colton, and R. Kingsbury, "Planet High Speed Radio: Crossing Gbps from a 3U CubeSat," in *Small Satellite Conference*, 2019.

[191] N. Saeed *et al.*, "CubeSat Communications: Recent Advances and Future Challenges," Aug. 26, 2019. arXiv: http://arxiv.org/abs/1908.09501v1 [eess.SP].

[192] R. Liu *et al.*, "An Analytical Framework for Resource-Limited Small Satellite Networks," *IEEE Communications Letters*, vol. 20, no. 2, pp. 388–391, Feb. 2016.

[193] M. Marchese and F. Patrone, "Energy-aware Routing Algorithm for DTN-Nanosatellite Networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2018.

[194] J. A. R. de Azua, A. Calveras, and A. Camps, "Internet of Satellites (IoSat): Analysis of Network Models and Routing Protocol Requirements," *IEEE Access*, vol. 6, pp. 20 390–20 411, 2018.

[195] Z. Liu, J. Zhu, J. Zhang, and Q. Liu, "Routing algorithm design of satellite network architecture based on SDN and ICN," *International Journal of Satellite Communications and Networking*, vol. 38, no. 1, pp. 1–15, Mar. 2019.

[196] Y. Zhu, L. Qian, L. Ding, F. Yang, C. Zhi, and T. Song, "Software defined routing algorithm in LEO satellite networks," in *2017 International Conference on Electrical Engineering and Informatics (ICELTICs)*, Oct. 2017.

[197] Q. Guo *et al.*, "SDN-Based End-to-End Fragment-Aware Routing for Elastic Data Flows in LEO Satellite-Terrestrial Network," *IEEE Access*, vol. 7, pp. 396–410, 2019.

[198] E. Ekici, I. Akyildiz, and M. Bender, "Datagram routing algorithm for LEO satellite networks," in *2000 IEEE International Conference on Computer Communications (INFOCOM)*, Mar. 2000.

[199] L. C. Freeman, "A Set of Measures of Centrality Based on Betweenness," *Sociometry*, vol. 40, no. 1, p. 35, Mar. 1977.

[200] J. Aspnes *et al.*, "On-line routing of virtual circuits with applications to load balancing and machine scheduling," *Journal of the ACM*, vol. 44, no. 3, pp. 486–504, May 1997.

[201] H. Kaushal and G. Kaddoum, "Optical Communication in Space: Challenges and Mitigation Techniques," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 57–96, 2017.

[202] W. Chen, J. Sun, X. Hou, R. Zhu, P. Hou, Y. Yang, M. Gao, L. Lei, K. Xie, M. Huang, R. Li, H. Zang, Y. Wan, E. Dai, Y. Xi, W. Lu, S. Wei, L. Liu, and J. Li, "5.12gbps optical communication link between LEO satellite and ground station," in *2017 IEEE International Conference on Space Optical Systems and Applications (ICSOS)*, Nov. 2017.

[203] S. Chaudhary, X. Tang, A. Sharma, B. Lin, X. Wei, and A. Parmar, "A cost-effective 100 gbps SAC-OCDMA–PDM based inter-satellite communication link," *Optical and Quantum Electronics*, vol. 51, no. 5, Apr. 2019.

[204] I. del Portillo, M. Sanchez, B. Cameron, and E. Crawley, "Optimal location of optical ground stations to serve LEO spacecraft," in *2017 IEEE Aerospace Conference*, Mar. 2017.

[205] N. K. Lyras, C. N. Efrem, C. I. Kourogiorgas, A. D. Panagopoulos, and P.-D. Arapoglou, "Optimizing the Ground Network of Optical MEO Satellite Communication Systems," *IEEE Systems Journal*, pp. 1–9, 2019.

[206] S. Nie, J. M. Jornet, and I. F. Akyildiz, "Deep-Learning-Based Resource Allocation for Multi-Band Communications in CubeSat Networks," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2019.

[207] Y. Drif, E. Chaput, E. Lavinal, P. Berthou, B. T. Jou, O. Gremillet, and F. Arnal, "An extensible network slicing framework for satellite integration into 5G," *International Journal of Satellite Communications and Networking*, Nov. 2020.

[208] T. Ahmed, A. Alleg, R. Ferrus, and R. Riggio, "On-Demand Network Slicing using SDN/NFV-enabled Satellite Ground Segment Systems," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, Jun. 2018.

[209] I. Bisio, F. Lavagetto, G. Verardo, and T. de Cola, "Network Slicing Optimization for Integrated 5G-Satellite Networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2019.

[210] T. de Cola and I. Bisio, "QoS Optimisation of eMBB Services in Converged 5G-Satellite Networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 12 098–12 110, Oct. 2020.

[211] J. Li, H. Lu, K. Xue, and Y. Zhang, "Temporal Netgrid Model-Based Dynamic Routing in Large-Scale Small Satellite Networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 6009–6021, Jun. 2019.

[212] D. M. Michal Pioro, *Routing, Flow, and Capacity Design in Communication and Computer Networks*. ACADEMIC PR INC, Jun. 1, 2004, 800 pp., ISBN: 0125571895.

[213] A. Alleg, T. Ahmed, M. Mosbah, R. Riggio, and R. Boutaba, "Delay-aware VNF placement and chaining based on a flexible resource allocation approach," in *2017 13th International Conference on Network and Service Management (CNSM)*, Nov. 2017.

[214] A. Tadanki and E. G. Lightsey, "Closing the Power Budget Architecture for a 1U CubeSatFramework," Georgia Institute of Technology, Tech. Rep., 2019.

[215] F. Alagoz and G. Gur, "Energy Efficiency and Satellite Networking: A Holistic Overview," *Proceedings of the IEEE*, vol. 99, no. 11, pp. 1954–1979, Nov. 2011.

[216] C. Zhan, H. Hu, X. Sui, Z. Liu, and D. Niyato, "Completion Time and Energy Optimization in the UAV-Enabled Mobile-Edge Computing System," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7808–7822, Aug. 2020.

[217] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, and P. Francois, "A Declarative and Expressive Approach to Control Forwarding Paths in Carrier-Grade Networks," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '15*, Aug. 2015.

[218] "IBM ILOG CPLEX Optimization Studio: CPLEX User's Manual," International Business Machines Corporation (IBM), Tech. Rep., 2020.

[219] Y. T. Hou, Y. Shi, and H. D. Sherali, *Applied Optimization Methods for Wireless Networks*. Cambridge, United Kingdom: Cambridge University Press, 2014, ISBN: 9781139088466.

[220] N. Feamster and J. Rexford, "Why (and How) Networks Should Run Themselves," in *Proceedings of the Applied Networking Research Workshop*, Jul. 2018.

[221] F. Kaltenberger, A. P. Silva, A. Gosain, L. Wang, and T.-T. Nguyen, "OpenAirInterface: Democratizing innovation in the 5G Era," *Computer Networks*, vol. 176, p. 107 284, Jul. 2020.

[222] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "srsLTE: an open-source platform for LTE evolution and experimentation," in *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization - WiNTECH '16*, 2016.

[223] Net4sat Development Community, *The OpenSAND Platform*, Apr. 2020.

VITA

Ahan Kak was born on October 15, 1994, in Mumbai, India. He received the Bachelor of Science degree in Electrical Engineering from the Veermata Jijabai Technological Institute, University of Mumbai, Mumbai, India, in 2016. He received the Master of Science and Doctor of Philosophy degrees in Electrical and Computer Engineering from the Georgia Institute of Technology, Atlanta, USA, in December 2020, and May 2021, respectively. His doctoral work was conducted under the supervision of Dr. Ian F. Akyildiz at the Broadband Wireless Networking Laboratory. He is a member of the IEEE and the ACM. His research interests include software-defined networking, network function virtualization, cellular networks, nanosatellite networks, and the Internet of Things.